ANALISIS SENTIMEN FILM PADA TWITTER BERBAHASA INDONESIA MENGGUNAKAN *ENSEMBLE FEATURES* DAN *NAÏVE BAYES*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rosy Indah Permatasari
NIM: 145150201111032

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS SENTIMEN FILM PADA TWITTER BERBAHASA INDONESIA MENGGUNAKAN *ENSEMBLE FEATURES* DAN *NAÏVE BAYES*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rosy Indah Permatasari
NIM: 145150201111032

Skripsi ini telah diuji dan dinyatakan lulus pada:
18 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
M. Ali Fauzi, S.Kom, M.Kom	Putra Pandu Adikara, S.Kom, M.Kom
NIK: 201502 890101 1 001	NIP: 19850725 200812 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Juli 2018

Rosy Indah Permatasari
NIM: 145150201111032

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis sentimen film pada Twitter berbahasa Indonesia menggunakan *Ensemble Features* dan Naïve Bayes".

Dalam penulisan skripsi ini, penulis mendapatkan banyak dukungan dan bantuan maupun material dari berbagai pihak. Oleh karena itu penulis mengucapkan banyak terima kasih kepada:

1. Bapak M. Ali Fauzi, S.Kom, M.Kom dan Bapak Putra Pandu Adikara, S.Kom, M.Kom selaku dosen Pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
2. Bapak Agus Wahyu Widodo, S.T., M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak M. Tanzil Furqon, S.Kom., M.Comp.Sc dan Ibu Indriati, S.T, M.Kom, selaku dosen Penasihat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh studi.
5. Kedua orang tua penulis, Roelly Prastyawati dan H.R. Soenardjo, S.H., Om Handoko, Tante Richa, serta keluarga, yang telah memberi dukungan baik moril ataupun materil, motivasi, serta doa demi kelancaran pengerjaan skripsi ini.
6. Ibu Eka Dewi Lukmana Sari, M.Pd, selaku pakar yang telah bersedia menyediakan waktu untuk mevalidasi data skripsi saya.
7. Seluruh Dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberi ilmunya kepada penulis selama perkuliahan di kampus ini.
8. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi dan menyelesaikan skripsi di Teknik Informatika Universitas Brawijaya.
9. Mbak Umi Rofiqoh sebagai kakak seniorku yang telah membantuku dalam pengerjaan drama skripsi ini.
10. Teman-teman seperjuangan yang telah banyak membantu dalam drama skripsi penulis baik secara langsung maupun tidak langsung yaitu Azmi Makarima, Fachrul Rozy S.R., Mbak Resti Febri, Ema Agasta, Dwi Wahyu Puji L., Icha Indrayana, Anggi Wiyani Putri, Amalia Kusuma A., Fakhruddin Farid I., Dyah Ayu W. D., dan yang lainnya.
11. Teman-teman AIESEC Atmosphere yaitu Husna F. Ridharti dan Auli Nafisa.

Semoga Allah SWT senantiasa membalas segala kebaikan dan jasa yang telah diberikan kepada penulis. Pada intinya, skripsi ini membahas tentang sistem berupa analisis sentimen yang mampu menerapkan *Ensemble Features* dan Naïve Bayes untuk mengklasifikasikan opini positif dan negatif Film pada Twitter berbahasa Indonesia untuk mengetahui pengaruh penggunaan ensemble features terhadap hasil analisis sentimen opini film Indonesia pada Twitter

Penulis menyadari bahwa skripsi ini memiliki banyak kekurangan baik pada format penulisan maupun isi bahasannya, oleh karena itu diharapkan saran dan kritik yang dapat membangun. Semoga skripsi ini dapat memberikan manfaat bagi semua pihak baik untuk pembaca maupun penelitian selanjutnya.

Malang, 18 Juli 2018

Penulis
rosyindahp@student.ub.ac.id

ABSTRAK

Rosy Indah Permatasari, Analisis Sentimen Film pada Twitter Berbahasa Indonesia menggunakan *Ensemble Features* dan *Naïve Bayes*.

Pembimbing: M. Ali Fauzi S.Kom., M.Kom dan Putra Pandu Adikara S.Kom., M.Kom.,

Analisis sentimen atau *opinion mining* merupakan salah satu topik penelitian terkini di bidang pengolahan informasi yang bertujuan untuk mengetahui apakah polaritas sebuah data yang berbentuk teks (dokumen, kalimat, paragraf) akan mengarah pada bersifat positif, negatif, ataupun netral. Dokumen teks yang digunakan dalam penelitian berasal dari Twitter tentang opini film berbahasa Indonesia. Twitter memiliki banyak fitur yang mana salah satunya dapat dimanfaatkan sebagai sarana untuk mengetahui opini film dari seluruh pengguna secara umum tidak terbatas oleh pecinta film saja. Dalam hal ini analisis sentimen mempunyai peranan penting untuk menunjang sarana tersebut. Metode yang digunakan adalah Naïve Bayes dengan menggunakan *Ensemble Features* sebagai pembaharuan fitur selain *Bag of Words Features*. Jenis-jenis *Ensemble Features* antara lain *Twitter specific features*, *textual features*, *part of speech features,* dan *lexicon based features*. Data yang digunakan pada penelitian ini sebanyak 500 data yang dibagi menjadi dua jenis data dengan perbandingan 70% untuk data latih dan 30% untuk data uji. Hasil akurasi sistem diperoleh dari analisis sentimen dengan metode Naïve Bayes dan *Ensemble Features* tanpa *bag of words features* sebesar 61,33%, *precision* sebesar 0,6308, *recall* sebesar 0,5467 dan *f-measure* sebesar 0,5857. Sedangkan hasil akurasi sistem menggunakan *Ensemble Features* dan *Bag of Words Features* sebesar 88,67 %, *precision* sebesar 0,9143, *recall* sebesar 0,8583, dan *f-measure* sebesar 0,8828.

Kata kunci: *analisis sentimen, Twitter, Naïve Bayes, Ensemble Features, opini film*

ABSTRACT

Rosy Indah Permatasari, *Sentiment Analysis Movie Review* in Twitter *Based on Indonesian Language Using Ensemble Features and Naïve Bayes*.

***Supervisors*: M. Ali Fauzi S.Kom., M.Kom and Putra Pandu Adikara S.Kom., M.Kom.,**

Sentiment analysis or opinion mining is one of the latest research topics in the field of information processing. It aims to know whether the polarity of a text-shaped data (document, sentence, paragraph) will lead to positive, negative, or neutral trait. This research used document text about Indonesian movie review which was obtained from Twitter. Twitter has many features in which one of it can be used as a media to know public opinion about movie, not only limited to movie lovers. In this case, sentiment analysis has an important role to support the media. The method used in this research was Naïve Bayes using *Ensemble Features* as a renewal feature beside *Bag of Words Features*. There are several types of *Ensemble Features* which are *Twitter specific features*, *textual features*, *part of speech features*, and *lexicon based features*. 500 data were used in this research, which were later divided into two types of data with the comparison of 70% for training data and 30% for testing data. The result of system accuracy obtained from sentiment analysis with Naïve Bayes and *Ensemble Features* without *bag of words features* is 61.33%, 0.6308 *precision*, 0.5467 *recall*, and 0.5857 *f-measure*. The result of system accuracy using *Ensemble Features* and *Bag of Words Features* is 88.67%, 0.9143 *precision*, 0.8533 *recall*, and 0.8828 *f-measure*.

Keywords: *sentiment analysis, Twitter, Naïve Bayes, Ensemble Features, movie review*

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xii
DAFTAR GAMBAR xiv
DAFTAR KODE PROGRAM xv
DAFTAR LAMPIRAN xvii
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 3
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Tinjauan Pustaka 5
2.2 Twitter 7
2.3 Opini Film 7
2.4 *Text Mining* 7
2.5 Analisis Sentimen 8
2.6 *Text Preprocessing* 8
2.6.1 Tokenisasi 8
2.6.2 *Data Cleaning* 9
2.6.3 *Case Folding* 9
2.6.4 *Filtering* atau *Stopword Removal* 10
2.6.5 *Stemming* 10
2.7 *Ensemble Features* 11

2.8 *Vector Space Model* 13
2.8.1 *Term Frequency* 13
2.9 *Naïve Bayes Classifier* 13
2.9.1 Gaussian Naïve Bayes 14
2.9.2 Bernoulli Naïve Bayes 14
2.9.3 Multinomial Naïve Bayes 15
2.10 Pengujian 15
BAB 3 METODOLOGI 18
3.1 Tipe Penelitian 18
3.2 Strategi Penelitian 18
3.3 Partisipan Penelitian 18
3.4 Lokasi Penelitian 18
3.5 Teknik Pengumpulan Data 19
3.6 Teknik Analisis Data 19
3.7 Perancangan Algoritme 19
3.8 Jadwal Penelitian 19
BAB 4 PERANCANGAN DAN IMPLEMENTASI 21
4.1 Tahapan-tahapan Utama Program 21
4.1.1 Tahapan pada Ensemble Features 22
4.1.2 Tahapan pada *Bag of Word Features* dan *Raw Term Frequency* 29
4.1.3 *Text Preprocessing* 30
4.1.4 Klasifikasi *Naïve Bayes* 37
4.2 Manualisasi 50
4.2.1 Manualisasi *Ensemble Features* 51
4.2.2 Manualisasi Klasifikasi *Naïve Bayes* 57
4.3 Perancangan Pengujian 69
4.3.1 Perancangan Pengujian Pengaruh *Twitter Specific Features* terhadap Akurasi Sistem 69
4.3.2 Perancangan Pengujian Pengaruh *Textual Features* terhadap Akurasi Sistem 70
4.3.3 Perancangan Pengujian Pengaruh *Part of Speech* (*PoS*) *Features* terhadap Akurasi Sistem 70

4.3.4 Perancangan Pengujian Pengaruh *Lexicon Based Features* terhadap Akurasi Sistem 70
4.3.5 Perancangan Pengujian Pengaruh *Bag of Words Features* terhadap Akurasi Sistem 70
4.3.6 Perancangan Pengujian Pengaruh *Ensemble Features* tanpa *Bag of Words features* terhadap Akurasi Sistem 70
4.3.7 Perancangan Pengujian Pengaruh *Ensemble Features* dan *Bag of Words Features* terhadap Akurasi Sistem 71
4.4 Implementasi *Ensemble Features* 71
4.4.1 Implementasi *Twitter Specific Features* 71
4.4.2 Implementasi *Textual Features* 73
4.4.3 Implementasi *Part of Speech Features* 78
4.4.4 Implementasi *Lexicon Based Features* 80
4.5 Implementasi *Bag of Words Features* 83
4.5.1 Implementasi *Case Folding* 84
4.5.2 Implementasi *Data Cleaning* 84
4.5.3 Implementasi *Stemming* 85
4.5.4 Implementasi *Filtering* 85
4.5.5 Implementasi Tokenisasi 86
4.5.6 Implementasi *Raw Term Frequency* 86
4.6 Implementasi *Naïve Bayes* 87
4.6.1 Implementasi *Prior* 87
4.6.2 Implementasi *Bernoulli Training* 88
4.6.3 Implementasi *Bernoulli Testing* 89
4.6.4 Implementasi *Gaussian Naive Bayes* 90
4.6.5 Implementasi *Multinomial Training* 92
4.6.6 Implementasi *Multinomial Testing* 93
4.6.7 Implementasi *Posterior* 94
4.7 Implementasi Pengujian 95
4.7.1 Implementasi Perhitungan *Confusion Matrix*. 95
4.7.2 Implementasi Perhitungan *Accuracy* 96
4.7.3 Implementasi Perhitungan *Precision* 96
4.7.4 Implementasi Perhitungan *Recall* 97

4.7.5 Implementasi Perhitungan *F-Measure* 97
4.8 Tampilan Program Sistem 97
4.8.1 Tampilan Program Input Data Uji 97
4.8.2 Tampilan Program Proses Pembuatan *Ensemble Features* dan *Bag of Word Features* 98
4.8.3 Tampilan Program Perhitungan Naive Bayes 98
4.8.4 Tampilan Program Pengujian 99
BAB 5 HASIL DAN PEMBAHASAN 100
5.1 Pengujian Pengaruh *Twitter Specific Features* 100
5.2 Pengujian Pengaruh *Textual Features* 101
5.3 Pengujian Pengaruh *Part of Speech Features* 102
5.4 Pengujian Pengaruh *Lexicon Based Features* 103
5.5 Pengujian Pengaruh *Bag of Words Features* 104
5.6 Pengujian Pengaruh *Ensemble Features* tanpa *Bag of Words* 105
5.7 Pengujian Pengaruh *Ensemble Features* dan *Bag of Words Features* 106
BAB 6 PENUTUP 109
6.1 Kesimpulan 109
6.2 Saran 110
DAFTAR PUSTAKA 111
LAMPIRAN 113

DAFTAR TABEL

Tabel 2.1 Kajian Pustaka 6
Tabel 2.2 Tahapan pada Tokenisasi 9
Tabel 2.3 Tahapan pada *Data Cleaning* 9
Tabel 2.4 Tahapan pada *Case Folding* 10
Tabel 2.5 Tahapan pada *Filtering* atau *Stopword Removal* 10
Tabel 2.6 Tahapan pada *Stemming* 11
Tabel 2.7 Daftar *Ensemble Features* 11
Tabel 2.8 Daftar *Ensemble Features* (lanjutan) 12
Tabel 2.9 *Confusion Matrix* 15
Tabel 3.1 Jadwal Penelitian 20
Tabel 4.1 Data Latih 51
Tabel 4.2 Data Uji 51
Tabel 4.3 Perhitungan *Twitter Specific Features* 52
Tabel 4.4 Perhitungan *Textual Features* 52
Tabel 4.5 Perhitungan *Part of Speech Features* 53
Tabel 4.6 Perhitungan *Lexicon Based Features* 54
Tabel 4.7 Hasil *Case Folding* 55
Tabel 4.8 Hasil *Data Cleaning* 55
Tabel 4.9 Hasil *Stemming* 55
Tabel 4.10 Hasil *Filtering* atau *Stopword Removal* 56
Tabel 4.11 Hasil Tokenisasi 56
Tabel 4.12 Perhitungan *Raw Term Frequency* 56
Tabel 4.13 Daftar *Ensemble Features* dengan *Bernoulli Naïve Bayes* 58
Tabel 4.14 Perhitungan Probabilitas Data Latih dengan *Bernoulli Naïve Bayes* ... 58
Tabel 4.15 Perhitungan Probabilitas Data Uji dengan *Bernoulli Naïve Bayes* 59
Tabel 4.16 Daftar *Ensemble Features* dengan *Multinomial Naïve Bayes* 59
Tabel 4.17 Perhitungan Probabilitas Data Latih dengan *Multinomial Naïve Bayes* 62
Tabel 4.18 Perhitungan Probabilitas Data Uji dengan *Multinomial Naïve Bayes*. 64
Tabel 4.19 Daftar *Ensemble Features* dengan *Gaussian Naïve Bayes* 66

Tabel 4.20 *Probability Density Function* pada *Gaussian Naïve Bayes* 67
Tabel 4.21 Perancangan Pengujian 69
Tabel 5.1 Hasil Pengujian Twitter *Specific Features* 100
Tabel 5.2 Hasil Pengujian *Textual Features* 101
Tabel 5.3 Hasil Pengujian *Part of Speech Features* 102
Tabel 5.4 Hasil Pengujian *Lexicon based Features* 103
Tabel 5.5 Hasil Pengujian *Bag of Words Features* 104
Tabel 5.6 Hasil Pengujian *Ensemble Features* tanpa *Bag of Words* 105
Tabel 5.7 Hasil Pengujian *Ensemble Features dan Bag of Words Features* 106

DAFTAR GAMBAR

Gambar 4.1 Tahapan-tahapan Utama pada Program 21
Gambar 4.2 Tahapan pada *Ensemble Features* 23
Gambar 4.3 Tahapan pada *Twitter Specific* 24
Gambar 4.4 Tahapan pada *Textual Features* 26
Gambar 4.5 Tahapan pada *Part of Speech Features* 28
Gambar 4.6 Tahapan pada *Lexicon Based Features* 29
Gambar 4.7 Tahapan pada *Bag of Words Features* 30
Gambar 4.8 Tahapan pada *Text Preprocessing* 31
Gambar 4.9 Tahapan pada *Case Folding* 32
Gambar 4.10 Tahapan pada *Data Cleaning* 33
Gambar 4.11 Tahapan pada *Stemming* 34
Gambar 4.12 Tahapan pada *Filtering* atau *Stopword Removal* 35
Gambar 4.13 Tahapan pada Tokenisasi 36
Gambar 4.14 Proses *Raw Term Frequency* 37
Gambar 4.15 Tahapan pada Klasifikasi *Naïve Bayes* 38
Gambar 4.16 Tahapan pada Naïve Bayes *Training* 39
Gambar 4.17 Tahapan pada Naïve Bayes *Testing* 40
Gambar 4.18 Tahapan pada *Bernoulli Naïve Bayes* untuk Data Latih 42
Gambar 4.19 Tahapan pada *Bernoulli Naïve Bayes* untuk Data Uji 43
Gambar 4.20 Tahapan pada *Gaussian Naïve Bayes* 46
Gambar 4.21 Tahapan pada *Multinomial Naïve Bayes* untuk Data Latih 48
Gambar 4.22 Tahapan pada *Multinomial Naïve Bayes* untuk Data Uji 49
Gambar 4.23 Cuplikan isi *file* stopword_list_tala.txt 86
Gambar 4.24 Tampilan Program Input Data Uji 98
Gambar 4.25 Tampilan Program Proses Pembuatan Ensemble Features dan Bag of Word Features 98
Gambar 4.26 Tampilan Program Perhitungan Naive Bayes 99
Gambar 4.27 Tampilan Program Pengujian 99
Gambar 5.1 Hasil Pengujian Seluruh Fitur 108

DAFTAR KODE PROGRAM

Kode Program 4.1 Implementasi F1 71
Kode Program 4.2 Implementasi F2 72
Kode Program 4.3 Implementasi F3 72
Kode Program 4.4 Implementasi F4 73
Kode Program 4.5 Implementasi F5 73
Kode Program 4.6 Implementasi F6 74
Kode Program 4.7 Implementasi F7 74
Kode Program 4.8 Implementasi F8 75
Kode Program 4.9 Implementasi F9 75
Kode Program 4.10 Implementasi F10 76
Kode Program 4.11 Implementasi F11 76
Kode Program 4.12 Implementasi F12 77
Kode Program 4.13 Implementasi FPOS 79
Kode Program 4.14 FPercentagePOS 79
Kode Program 4.15 Implementasi FLexicon 81
Kode Program 4.16 Implementasi FPercentageLexicon 82
Kode Program 4.17 Implementasi F37 83
Kode Program 4.18 Implementasi *Case Folding* 84
Kode Program 4.19 Implementasi *Data Cleaning* 84
Kode Program 4.20 Implementasi *Stemming* 85
Kode Program 4.21 Implementasi *Filtering* 85
Kode Program 4.22 Implementasi Tokenisasi 86
Kode Program 4.23 Implementasi *Raw Term Frequency* 87
Kode Program 4.24 Implementasi Prior 87
Kode Program 4.25 Implementasi Bernoulli Training 88
Kode Program 4.26 Implementasi Bernoulli Testing 89
Kode Program 4.27 Implementasi Gaussian Naive Bayes 91
Kode Program 4.28 Implementasi Multinominal Training 93
Kode Program 4.29 Implementassi Multinominal Testing 94
Kode Program 4.30 Implementasi Posterior 95

Kode Program 4.31 Implementasi Perhitungan *Confusion Matrix* 96
Kode Program 4.32 Implementasi Perhitungan *Accuracy* 96
Kode Program 4.33 Implementasi Perhitungan *Precision* 96
Kode Program 4.34 Implementasi Perhitungan *Recall* 97
Kode Program 4.35 Implementasi Perhitungan *F-Measure* 97

DAFTAR LAMPIRAN

Lampiran 1 Tabel Data Latih 113
Lampiran 2 Tabel Data Uji 114
Lampiran 3 Surat Kesediaan Pakar 115

BAB 1 PENDAHULUAN

Pada bab ini akan menjelaskan tentang latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan.

1.1 Latar belakang

Antusiasme masyarakat terhadap perfilman di Indonesia saat ini sudah semakin meningkat seiring dengan kualitas film yang semakin baik setiap tahunnya. Menurut Agung Sentausa, Ketua Bidang Fasilitasi Pembiayaan Film Badan Perfilman Indonesia (BPI), pada tahun 2015, rata-rata jumlah penonton untuk setiap film yang tayang adalah 133 ribu penonton. Di tahun 2016, jumlahnya bertambah menjadi 274 ribu penonton. Di tahun 2017, hingga akhir semester pertama atau hingga Juni 2017, rata-rata jumlah penonton per film mencapai 282 ribu penonton (MetroTV News, 2017). Dari data tersebut, terlihat antusiasme masyarakat Indonesia sudah semakin tinggi. Banyak *website* yang sudah menyediakan tentang ulasan film seperti Internet Movie Database (IMDb) dan Rotten Tomatoes. *Website-website* tersebut merupakan dua situs paling popular yang paling sering dikunjungi oleh konsumen untuk mendapatkan sebuah informasi terbaru mengenai opini film. Namun pengguna yang berada pada situs-situs tersebut hanya pengguna pecinta film saja, tidak meliputi seluruh pengguna secara umum. Di sisi lain, ada salah satu jejaring sosial yang bisa dimanfaatkan untuk mengetahui opini film yaitu Twitter.

Twitter adalah layanan *microblogging* jejaring sosial gratis yang memungkinkan anggotanya untuk mengirim pesan singkat yang biasa disebut *tweets* (Statista, 2017). Pada pertengahan 2016, Statista, salah satu perusahaan statistik terkemuka di internet telah merilis data negara pengguna Twitter terbesar di dunia. Indonesia menempati urutan ketiga terbesar di dunia dengan jumlah pengguna yaitu 24,3 juta pengguna Twitter. Berdasarkan data tersebut, tingginya jumlah pengguna Twitter di Indonesia dapat menjadi suatu sarana dalam memberikan ulasan, kritikan, saran, serta kesan pesan terhadap suatu film yang sudah ditonton. Ulasan, kritikan, saran, serta kesan pesan tersebut dapat digunakan sebagai bahan masukan untuk pembuat film Indonesia dalam mengembangkan film-film Indonesia yang lebih baik di masa depan, mengetahui tanggapan atau respon dari masyarakat terhadap film yang telah ditonton, serta sebagai pertimbangan masyarakat untuk film yang akan ditonton.

Twitter memiliki banyak fitur yang mana salah satunya dapat dimanfaatkan sebagai sarana untuk mengetahui opini film dari seluruh pengguna secara umum tidak terbatas oleh pecinta film saja. Dalam hal ini analisis sentimen mempunyai peranan penting untuk menunjang sarana tersebut. Analisis sentimen biasanya digunakan dalam menganalisis terhadap sebuah produk dalam meningkatkan kualitas produk yang akan datang. Analisis sentimen merupakan proses yang bertujuan untuk

mengetahui apakah polaritas sebuah data yang berbentuk teks (dokumen, kalimat, paragraf) akan mengarah pada bersifat positif, negatif, ataupun netral (Kontopoulos, Berberidis, Dergiades, & Bassiliades, 2013). Dalam hal ini analisis sentimen dapat diterapkan pada opini film Indonesia.

Penelitian sebelumnya yang berkaitan dengan analisis sentimen yang dilakukan oleh Agnes Rossi pada tahun 2017. Dalam penelitian ini metode yang digunakan adalah Naïve Bayes yang berfungsi untuk mengklasifikasikan sentimen positif dan negatif opini. Saat dilakukan pengujian dengan menggabungkan pembobotan tekstual dan non-tekstual menghasilkan akurasi 74,81% (Rossi, Lestari, Perdana, & Fauzi, 2017). Kemudian penelitian selanjutnya terkait dengan analisis sentimen adalah penelitian yang dilakukan oleh Perdana pada tahun 2013. Dalam penelitian ini membahas tentang pengkategorian pesan singkat berbahasa Indonesia menggunakan metode Naïve Bayes. Hasil pengujian didapatkan nilai *precision* sebesar 80%, *recall* sebesar 79%, dan *f-measure* sebesar 78% (Perdana, Suprapto, & Regasari, 2013).

Kemudian penelitian selanjutnya terkait dengan analisis sentimen yaitu dilakukan oleh Siddiqua, Ahsan, dan Chy pada tahun 2016. Pada penelitian ini dilakukan penggabungan *classifier* berbasis aturan dengan *Ensemble Features* dan analisis sentimen pada *microblog*. Selain itu, penggunaan semua fitur semantik jika digabungkan dengan fitur *Bag of Words* (*BoW*) akan menghasilkan akurasi sebesar 87,7%. (Siddiqua, Ahsan, & Chy, 2016). Selanjutnya penelitian terkait dengan analisis sentimen yaitu dilakukan oleh Pak dan Paroubek pada tahun 2010. Pada penelitian ini metode yang digunakan dalam membangun sebuah *sentiment classifier* adalah Naïve Bayes. Selain itu, peneliti juga mencoba Support Vector Machine (SVM), dan *Conditional Random Field* (CRF) sebagai *classifier*. Namun, hasil dari penelitian ini menunjukkan bahwa metode Naïve Bayes menghasilkan *classifier* terbaik (Patodkar & I.R, 2016).

Berdasarkan latar belakang dan hasil uraian dari penelitian-penelitian sebelumnya, maka melalui penelitian ini dilakukan analisis sentimen film pada Twitter berbahasa Indonesia yang mana metodenya menggunakan *Ensemble Features* dan Naïve Bayes. Penelitian ini diharapkan dapat menjadi bahan rujukan untuk konsumen dan sebagai bahan pengujian sebuah film untuk produser.

1.2 Rumusan masalah

Berdasarkan latar belakang di atas, maka penulis merumuskan pokok permasalahan terkait penelitian ini yaitu:

1. Bagaimana penggunaan *ensemble features* terhadap hasil analisis sentimen opini film Indonesia pada Twitter?
2. Terdapat hubungan positif antara penggunaan *ensemble features* dan performa klasifikasi.

1.3 Tujuan

Tujuan yang ingin dicapai oleh penulis dalam penelitian ini adalah untuk mengetahui pengaruh penggunaan *ensemble features* terhadap hasil analisis sentimen opini film Indonesia pada Twitter

1.4 Manfaat

Penelitian yang dilakukan diharapkan dapat memberikan manfaat bagi pihak-pihak berkepentingan, antara lain yaitu:

Manfaat bagi Konsumen adalah penelitian ini bisa menjadi sebagai bahan rujukan (*second opinion)* atau pertimbangan konsumen untuk menonton film yang akan ditonton.

Manfaat bagi Produser:

1. Penelitian ini bisa menjadi sebagai strategi marketing untuk film yang akan tayang dimana bisa dikategorikan umur maupun jenis kelamin.
2. Penelitian ini bisa menjadi bahan pengujian dan perbaikan sebuah film.
3. Penelitian ini bisa menjadi bahan untuk mengetahui opini masyarakat terhadap film apakah opini tersebut termasuk kategori positif atau negatif.

Manfaat Bagi Penulis:

1. Sebagai wadah untuk mengimplementasikan ilmu yang telah didapatkan terutama di bidang *text mining*.
2. Mendapatkan pemahaman mengenai penerapan metode *Ensemble Feature* dan Naïve Bayes untuk analisis sentimen film berbahasa Indonesia pada Twitter.
3. Menambah wawasan dalam bidang sentimen analisis.

1.5 Batasan masalah

Untuk memfokuskan penelitian yang akan dilakukan permasalahan yang dibatasi adalah sebagai berikut:

1. Metode yang digunakan untuk pengklasifikasian adalah Naïve Bayes tanpa membandingkan metode lain.
2. Kelas yang digunakan pada penelitian ini hanya dua yaitu positif dan negatif.
3. Sistem tidak memperhatikan kata tidak baku.
4. Dokumen teks yang digunakan pada penelitian ini menggunakan Bahasa Indonesia.
5. Data yang digunakan pada penelitian ini diambil dari *Twitter*.

6. Sistem tidak memperhatikan kalimat negasi.

1.6 Sistematika pembahasan

Dalam skripsi ini dibagi menjadi tujuh bab dengan beberapa subbab. Adapun sistematika pembahasan proposal skripsi adalah sebagai berikut:

BAB I **PENDAHULUAN**

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, ruang lingkup atau batasan masalah, dan sistematika pembahasan dari penelitian yang diangkat.

BAB II **LANDASAAN KEPUSTAKAAN**

Menguraikan dalam sub bab-sub bab terkait pokok-pokok teori yang terkait dengan implementasi.

BAB III **METODOLOGI PENELITIAN**

Bab ini menguraikan tentang metode atau langkah-langkah yang digunakan dalam penelitian ini.

BAB IV **PERANCANGAN DAN IMPLEMENTASI**

Bab ini menguraikan tentang perhitungan manualisasi program dan implementasi program.

BAB V **HASIL DAN PEMBAHASAN**

Bab ini menguraikan tentang pengujian program.

BAB VI **PENUTUP**

Bab ini menguraikan tentang kesimpulan dari keseluruhan hasil penelitian dan saran untuk penelitian yang selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini menjelaskan pembahasan mengenai review penelitian-penelitian yang terdahulu yang berkaitan dengan permasalahan yang diangkat pada penelitian ini. Selain itu, bab ini juga menguraikan teori-teori dasar didalam penelitian ini serta teori pendukung yang digunakan seperti film berbahasa Indonesia, text mining, Twitter, analisis sentimen, *preprocessing*, *Ensemble Features* dan metode yang akan digunakan yaitu Naïve Bayes serta pengujian.

2.1 Tinjauan Pustaka

Penelitian sebelumnya yang berkaitan dengan analisis sentimen yang dilakukan oleh Agnes Rossi (2017). Pada penelitian ini metode yang digunakan adalah Naïve Bayes yang berfungsi untuk mengklasifikasikan sentimen positif dan negatif opini pilkada Daerah Khusus Ibukota Jakarta 2017 pada dokumen *Twitter* Berbahasa Indonesia. Selain Naïve Bayes, pada penelitian ini menggunakan pembobotan emoji untuk mendapatkan hasil penambahan akurasi yang lebih baik. Hasil pengujian pada penelitian ini menunjukkan bahwa dari hasil akurasi sistem pada pembobotan tekstual saja yaitu 68,52% pada pembobotan non-tekstual saja yaitu 75,93% dan pada penggabungan kedua pembobotan yaitu 74,81% sehingga mampu membuktikan tingkat keberhasilan sistem dengan algoritme Naïve Bayes dalam mengklasifikasikan tweet yang mengandung sentimen positif ataupun negatif (Rossi et al., 2017).

Penelitian selanjutnya terkait dengan analisis sentimen adalah penelitian yang dilakukan oleh Perdana (2013). Pada penelitian metode klasifikasi yang digunakan adalah Naïve Bayes untuk mengkategorikan pesan singkat berbahasa Indonesia pada Jejaring Sosial Twitter. Hasil pengujian pada penelitian ini menunjukkan bahwa dengan menggunakan data latih yang diambil dari dokumen RSS menghasilkan nilai rata-rata *recall* (ukuran dari jumlah dokumen benar dari suatu kategori yang berhasil diklasifikasikan oleh sistem) sebesar 0.793295455 (79%), *precision* (ukuran dari jumlah dokumen yang diklasifikasikan oleh sistem dan dokumen tersebut benar) sebesar 0.804259992 (80%) dan F_1 *Measure* (gabungan dari nilai *precision* dan *recall*) sebesar 0.78436341 (78%). Pada penelitian ini fitur yang digunakan hanya *Bag of Words (BoW)* (Perdana et al., 2013).

Penelitian selanjutnya terkait dengan analisis sentimen yaitu dilakukan oleh Umme Aymun Siddiqua, Tanveer Ahsan, dan Abu Nowshed Chy pada tahun 2016. Pada penelitian ini dilakukan penggabungan *classifier* berbasis aturan dengan *Ensemble Features* dan teknik *machine learning* untuk analisis sentiment pada *microblog*. Hasil pengujian dari penelitian ini menunjukkan bahwa penggunaan *Ensemble Features* dapat meningkatkan akurasi dan mendapati akurasi tertinggi jika dibandingkan dengan menggunakan fitur secara terpisah. Pada penggunaan fitur *Bag of Words* saja hanya menghasilkan akurasi sebesar 73,8%. Penggunaan semua fitur

semantik jika digabungkan dengan fitur *Bag of Words* (*BoW*) akan menghasilkan akurasi sebesar 87,7%. Penggunaan *ensemble features* lebih baik dibandingkan dengan *unigram+bigram* yaitu dengan akurasi sebesar 83,3%, *label propagation* sebesar 84,7%, *sentiment topic features* sebesar 86,7%, *sentistrength* sebesar 81,7%, *meta level features* sebesar 81,6%, *semantria (online system)* sebesar 78,1% (Siddiqua et al., 2016).

Penelitian selanjutnya terkait dengan analisis sentimen yaitu dilakukan Alexander Pak dan Patrick Paroubek. Pada penelitian ini, peneliti memperlihatkan metode untuk sekumpulan data korpus yang dapat digunakan untuk melatih *sentiment classifier*. Dalam membangun sebuah *sentiment classifier* peneliti mencoba untuk menggunakan *Naïve Bayes*, *Support Vector Machine* (SVM), dan *Conditional Random Field* (CRF). Hasil dari penelitian ini menunjukkan bahwa metode Naïve Bayes menghasilkan *classifier* terbaik (Patodkar & I.R, 2016). Perbedaan terhadap ketiga penelitian sebelumnya dengan penelitian skripsi yang diajukan berdasarkan studi kepustakaan yang telah dilakukan akan ditunjukkan pada Tabel 2.1.

Tabel 0.1 Kajian Pustaka

No	Judul	Objek	Metode	Perbedaan Skripsi Peneliti
1	Analisis Sentimen Tentang Opini Pilkada DKI 2017 Pada Dokumen *Twitter* Berbahasa Indonesia Menggunakan Naïve Bayes dan Pembobotan Emoji (Rossi et al., 2017)	Analisis Sentimen pada Opini Pilkada DKI 2017	*Naive Bayes*	Perbedaan terdapat pada fitur yang digunakan yaitu hanya *Bag-of-Words* (*BoW)*
2	Pengkategorian Pesan Singkat Berbahasa Indonesia pada Jejaring Sosial *Twitter* dengan Metode Klasifikasi Naïve Bayes (Perdana et al., 2013)	Pesan Singkat Berbahasa Indonesia	*Naïve Bayes*	Perbedaan terdapat pada Fitur yang digunakan. Fitur didalam penelitian ini hanya menggunakan *Bag of Words (BoW)*
3	*Combining a Rule-based Classifier with Ensemble of Feature Sets and Machine Learning Techniques for Sentiment*	Analisis Sentimen pada Twitter	*Naive Bayes, SVM (Support Vector Machine)*	Perbedaan terdapat pada metode yang digunakan, yaitu *Naive Bayes*

	Analysis on Microblog (Siddiqua et al., 2016)			
4	*Twitter as a Corpus for Sentiment Analysis and Opinion Mining* (Pak & IParoubek, 2010)	Analisis sentimen pada Twitter	*Naïve Bayes*	Perbedaan pada penelitian ini hanya menggunakan *POS-tagging* sebagai fitur

Berdasarkan tinjauan pustaka di atas tersebut, maka diusulkan penelitian ini menggunakan *ensemble features* dan metode Naive Bayes.

2.2 Twitter

Twitter merupakan layanan *microblogging* terbesar didunia yang menawarkan fungsi weblog dan fungsi jejaring sosial. Twitter sejauh ini merupakan salah satu situs jejaring sosial *built-in* paling aktif. *Tweet* yang dibuat atau di *retweet* oleh pengguna akan muncul secara otomatis *timeline* pengguna yang diikutinya (Kwak, Lee, Park, & Moon, 2010). Twitter adalah salah satu bentuk *free microblogging* yang memungkinkan pengguna untuk mengirim dan menerima pesan singkat yang disebut *tweets*. *Tweets* dibatasi hingga 140 karakter dan dapat terhubung dengan *blogs,* halaman *web*, gambar, video, dan semua konten online lainnya. Pengguna dapat melakukan *tweeting* dalam 10 menit atau kapanpun dari komputer, ponsel, maupun tablet. Dengan mengikuti pengguna lain, pengguna dapat bertemu dengan sesama pengguna yang memiliki minat dan ketertarikan yang sama (Mollett, Moran, & Dunleavy, 2011).

2.3 Opini Film

Opini film merupakan studi, interpretasi, dan pengujian sebuah film. Opini film biasanya menawarkan sebuah makna, penilaian terhadap sebuah film dibandingkan dengan film lain, dan bisa memprediksi seberapa besar dampaknya terhadap konsumen. Sebuah opini dapat membahas film tertentu ataupun melihat sekumpulan film dan *genre* yang sama dan membantu untuk peningkatan kinerja sebuah produsen untuk membuat film yang lebih baik lagi dan mengetahui respon dari konsumen (ResearchGuides, 2017).

2.4 *Text Mining*

Text mining atau biasa dikenal dengan pemrosesan teks merupakan suatu bidang yang sedang berkembang dalam mengumpulkan suatu informasi penting dari teks bahasa. *Text mining* memiliki ciri-ciri sebagai proses proses menganalisis teks untuk mengekstrak informasi yang berguna untuk sebuah tujuan tertentu. Dalam masa sekarang, teks merupakan sarana yang paling umum dalam pertukaran

informasi secara formal. Bidang penambangan teks biasanya berhubungan dengan teks yang fungsinya adalah sebagai komunikasi informasi atau opini faktual (Tsai, 2011). Selain itu *text mining* atau biasa disebut juga penambangan teks dapat didefinisikan sebagai sebuah proses pengetahuan yang intensif yang mana pengguna dapat berinteraksi dalam melakukan pengumpulan dokumen dari waktu ke waktu. *Text mining* berusaha untuk mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi pola yang menarik (Feldman & Sanger, 2006).

Text mining berbeda dari yang dikenal dengan pencarian *web* biasa. Dalam hal pencarian, pengguna biasanya mencari sesuatu yang sudah diketahui dan sudah ditulis oleh orang lain. Masalahnya adalah bagaimana cara menyingkirkan materi yang saat ini tidak sesuai dengan kebutuhan pengguna untuk menemukan sebuah informasi yang relevan. Tujuan *text mining* sendiri adalah bagaimana menemukan informasi yang sebenarnya tidak diketahui, sesuatu yang belum diketahui oleh siapapun dan belum bisa dapat dituliskan (Hearst, 2003). Dalam berdasarkan teori-teori tersebut. *Text mining* dapat diaplikasikan dalam dunia nyata salah satunya adalah analisis sentimen.

2.5 Analisis Sentimen

Analisis sentimen atau *opinion mining* merupakan salah satu topik penelitian terkini di bidang pengolahan informasi. Teknik pengambilan informasi tekstual difokuskan pada pemrosesan, pencarian atau informasi *factual mining*. Ada unsur tekstual lain yang mengekspresikan karakter subjektif. Unsur-unsur ini terutama opini, sentimen, penilaian, sikap, dan emosi yang menjadi fokus sentimen analisis (Serrano-Guerrero, Olivas, Romero, & Herrera-Viedma, 2015). Selain itu, analisis sentimen merupakan proses yang bertujuan untuk mengetahui apakah polaritas sebuah data yang berbentuk teks (dokumen, kalimat, paragraf) akan mengarah pada bersifat positif, negatif, ataupun netral (Kontopoulos et al., 2013).

2.6 Text Preprocessing

Text preprocessing merupakan proses pembersihan dan penyiapan teks untuk klasifikasi. Teks *online* biasanya mengandung banyak *noise* dan bagian yang tidak informatif seperti *tag* HTML, *script,* dan iklan (Haddi, Liu, & Shi, 2013). Selain itu, *text preprocessing* merupakan bagian penting dari pemrosesan bahasa alami di mana karakter, kata, dan kalimat yang diidentifikasi pada tahap ini adalah unit yang mendasar yang nantinya akan diproses pada tahap selanjutnya. Data teks yang mengandung format khusus seperti format angka, format tanggal, dan kata yang

paling umum akan dipraproses oleh *text mining* sedangkan preposis dan kata ganti akan dihilangkan (Vijayarani, Ilamathi, & Nithya, 2015).

2.6.1 Tokenisasi

Tokenisasi merupakan proses pemecahan aliran teks ke dalam kata-kata, ungkapan, simbol, atau elemen bermakna lainnya yang disebut token. Tujuan dari tokenisasi adalah mengeksplorasi kata-kata dalam sebuah kalimat. Daftar token akan menjadi *input* tahapan selanjutnya seperti *parsing* atau *text mining*. Tokenisasi juga berguna dalam linguistik dan dalam ilmu komputer, tokenisasi menjadi bagian analisis leksikal (Vijayarani et al., 2015). Berikut ini adalah contoh tokenisasi dalam dokumen Bahasa Indonesia yang akan ditunjukkan pada Tabel 2.2.

Tabel 0.2 Tahapan pada Tokenisasi

Teks
”elearning di PTIIK di atas jam 6 sore kok selalu gak bisa dibuka ya?”
Hasil Token
“elearning”, “di”, “PTIIK”, ”di”, atas”, ”jam”, “6”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”, “?”

2.6.2 Data Cleaning

Data Cleaning merupakan sebuah tahapan dalam *text preprocessing* untuk menghapus beberapa komponen yang tidak penting pada sebuah dokumen teks (Manning, Raghavan, & Schütze, 2008). Komponen yang akan dihilangkan pada dokumen teks terdiri dari *username* (@), tautan *URL* (http atau https), *hashtag* (#), *retweet* (RT), tanda baca, angka, dan tanda baca lain yang tidak memiliki arti pada sebuah kalimat. Berikut ini adalah contoh *cleansing* yang akan ditunjukkan pada Gambar 2.3.

Tabel 0.3 Tahapan pada *Data Cleaning*

Hasil Token
“elearning”, “di”, “PTIIK”, ”di”, atas”, ”jam”, “6”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”, “?”
Hasil *Data Cleaning*
“elearning”, “di”, “PTIIK”, ”di”, atas”, ”jam”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”

2.6.3 Case Folding

Case Folding merupakan sebuah tahapan dalam *text preprocessing* untuk mengkonversi semua karakter huruf menjadi semua huruf kecil. Kumpulan kata-kata ini biasanya berawal dari kata yang sudah dikapitalisasi sebelumnya (Manning et al., 2008). Berikut ini adalah contoh *case folding* yang akan ditunjukkan pada Gambar 2.4.

Tabel 0.4 Tahapan pada *Case Folding*

Hasil *Data Cleaning*
“elearning”, “di”, “PTIIK”, ”di”, atas”, ”jam”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”
Hasil *Case Folding*
“elearning”, “di”, “ptiik”, ”di”, atas”, ”jam”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”

2.6.4 *Filtering* atau *Stopword Removal*

Kata-kata *stopword* sering digunakan dalam penggunaan kata-kata umum seperti ‘dan’, ‘adalah’, ‘ini’ dan sebagainya. Kata-kata tersebut tidak mempunyai makna yang berarti dalam pengklasifikasian dokumen sehingga kata-kata tersebut harus dihilangkan. Namun, perkembangan daftar kata-kata *stopword* semacam itu susah dan tidak konsisten antar sumber teks. Proses ini juga mengurangi data teks dan memperbaiki kinerja (Vijayarani et al., 2015). *Filtering* adalah tahap pemilihan kata-kata penting dari hasil token, yaitu kata-kata apa saja yang akan digunakan untuk mewakili dokumen. Berikut ini adalah contoh penggunaan *filtering* atau *stopword removal* dalam dokumen Bahasa Indonesia yang akan ditunjukkan pada Tabel 2.5.

Tabel 0.5 Tahapan pada *Filtering* atau *Stopword Removal*

Hasil *Case Folding*
“elearning”, “di”, “ptiik”, ”di”, atas”, ”jam”, “sore”, ”kok”, ”selalu”, ”gak”,”bisa”, “buka”, “ya”
Hasil *Filtering* atau *Stopword*

“elearning”, “ptiik”, “atas”, “jam”, “sore”, “selalu”, “bisa”, “buka”

2.6.5 *Stemming*

Proses *stemming* merupakan proses mengubah atau menguraikan bentuk kata menjadi kata dasar atau tahapan mencari dasar kata dari tiap kata hasil dari *filtering* atau *stopword removal*. *Stemming* biasanya mengacu pada proses heuristik kasar yang memotong ujung kata dengan harapan dapat mencapai tujuan tersebut (Manning et al., 2008). Dibawah ini akan ditunjukkan contoh penggunaan *stemming* pada Dokumen Bahasa Indonesia pada Tabel 2.6.

Tabel 0.6 Tahapan pada *Stemming*

Hasil *Filtering* atau *Stopword Removal*
“elearning”, “ptiik”, “atas”, “jam”, “sore”, “selalu”, “bisa”, “buka”
Hasil *Stemming*
“elearning”, “ptiik”, “atas”, “jam”, “sore”, “selalu”, “bisa”, “buka”

2.7 *Ensemble Features*

Ensemble Features merupakan gabungan dari beberapa fitur yang mematuhi aturan-aturan yang sudah ditetapkan. Berikut ini akan ditunjukkan beberapa fitur yang akan digunakan pada penelitian ini pada Tabel 2.7.

Tabel 0.7 Daftar *Ensemble Features*

Tipe	**ID**	**Deskripsi Fitur**
Twitter Specific	F1	Apakah *tweet* mengandung *#hashtag* atau tidak
	F2	Apakah *tweet* mengandung *retweet* atau tidak
	F3	Apakah *tweet* mengandung *username* atau tidak
	F4	Apakah *tweet* mengandung URL atau tidak
Textual Features	F5	Panjang Tweet: Jumlah kata dalam *tweet*
	F6	Rata-rata panjang kata: Rata-rata panjang karakter sebuah kata
	F7	Jumlah tanda tanya yang terdapat pada sebuah *tweet*
	F8	Jumlah tanda seru yang terdapat pada sebuah *tweet*

	F9	Jumlah tanda kutip yang terdapat pada sebuah *tweet*
	F10	Jumlah kata yang diawali dengan huruf besar pada *tweet*
	F11	Apakah *tweet* mengandung positif emoticon atau tidak
	F12	Apakah *tweet* mengandung negatif emoticon atau tidak
Parts of Speech (PoS) *Features*	F13	Jumlah kata benda POS yang tersedia dalam *tweet*
	F14	Jumlah kata sifat POS yang tersedia dalam *tweet*
	F15	Jumlah kata kerja POS yang tersedia dalam *tweet*
	F16	Jumlah kata keterangan POS yang tersedia dalam *tweet*
	F17	Jumlah kata penghubung POS yang tersedia dalam *tweet*

Tabel 0.8 Daftar *Ensemble Features* (lanjutan)

Tipe	**ID**	**Deskripsi Fitur**
Parts of Speech (PoS) *Features*	F18	Persentase kata benda POS yang tersedia dalam *tweet*
	F19	Persentase kata sifat POS yang tersedia dalam *tweet*
	F20	Persentase kata kerja POS yang tersedia dalam *tweet*
	F21	Persentase kata keterangan POS yang tersedia dalam *tweet*
	F22	Persentase kata penghubung POS yang tersedia dalam *tweet*
Lexicon Based Features	F23	Jumlah kata positif yang tersedia pada sebuah *tweet*
	F24	Jumlah kata negatif yang tersedia pada sebuah *tweet*
	F25	Jumlah kata positif dengan kata sifat POS
	F26	Jumlah kata negatif dengan kata sifat POS
	F27	Jumlah kata positif dengan kata kerja POS
	F28	Jumlah kata negatif dengan kata kerja POS
	F29	Jumlah kata positif dengan kata keterangan POS
	F30	Jumlah kata negatif dengan kata keterangan POS
	F31	Persentase kata positif dengan kata sifat POS
	F32	Persentase kata negatif dengan kata sifat POS

	F33	Persentase kata positif dengan kata kerja POS
	F34	Persentase kata negatif dengan kata kerja POS
	F35	Persentase kata positif dengan kata keterangan POS
	F36	Persentase kata negatif dengan kata keterangan POS
	F37	Jumlah kata *intensifier* yang tersedia pada sebuah *tweet*

Sumber: (Siddiqua, et al., 2016)

Fitur-fitur di atas akan dikelompokkan menjadi beberapa kategori fitur yaitu, *twitter specific features, textual features, Part of Speech* (PoS) *features, lexicon based features,* dan *Bag of Words.* Pada bagian *emoticon* akan digunakan kamus *emoticon*[1]. Fitur menggunakan *#hashtag* dan *retweet* untuk mengambil nilai pada fitur spesifik (*twitter specific*). Pada tipe *textual features,* pengambilan nilai fitur yang dilakukan hanya berdasarkan informasi eksplisit yang terdapat di dalam teks, seperti jumlah kata, jumlah kemunculan tanda petik atau tanda seru. Pada bagian *lexicon based features* menggunakan bantuan *sentiwords*[2] dan *booster words* [3]untuk pengambilan nilai fitur Dan pada bagian fitur *Bag of Words* akan dilakukan pembobotan *term frequency inverse document frequency* (Siddiqua et al., 2016).

2.8 *Vector Space Model*

Vector Space Model (VSM) adalah representasi sekumpulan dokumen sebagai vektor dalam ruang vektor umum dan sangat penting bagi sejumlah operasi pencarian informasi dimulai dari mencetak informasi pada sebuah *query,* klasifikasi dokumen, dan pengelompokan dokumen (Manning et al., 2008).

2.8.1 *Term Frequency*

tf adalah *term frequency*, dan $tf_{i,j}$ adalah banyaknya kemunculan *term ti* dalam dokumen *dj*, *term frequency* (*tf*) dihitung dengan menghitung banyaknya kemunculan term *ti* dalam dokumen *dj*. Rumus perhitungan *term frequency* akan ditunjukkan pada Persamaan 2.1 (Robertson, 2004).

$$Tf = tfij \quad (2.1)$$

1.8.1.1 *Raw Term Frequency*

Raw Term Frequency menganggap semua *term* dianggap sama pentingnya dalam hal menilai relevansi pada kueri. Sebenarnya beberapa *term* tertentu hanya sedikit atau bahkan tidak memiliki kekuatan dalam menentukan relevansinya

[1] https://github.com/masdevid/sentistrength_id/blob/master/emoticon_id.txt
[2] https://github.com/masdevid/sentistrength_id/blob/master/sentiwords_id.txt
[3] https://github.com/masdevid/sentistrength_id/blob/master/boosterwords_id.txt

(Manning et al., 2008). Sebagai contoh, sebuah *term* yang muncul 5 kali pada sebuah dokumen kenyataannya lebih penting dalam merepresentasikan sebuah dokumen dibandingkan *term* yang muncul satu kali. Namun, bukan berarti 5 kali lebih penting daripada yang muncul satu kali.

2.9 *Naïve Bayes Classifier*

Naïve Bayes Classifier merupakan klasifikasi statistik dengan tujuan memprediksi probabilitas keanggotaan kelas, seperti probabilitas sampel yang diberikan termasuk dalam kelas tertentu. Klasifikasi Bayesian didasarkan pada teorema Bayes. Klasifikasi Naïve Bayes mengasumsikan bahwa nilai atribut pada kelas tertentu tidak bergantung pada nilai atribut lainnya. Klasifikasi naïve bayes mempunyai kelebihan yaitu hanya membutuhkan sedikit jumlah pelatihan data untuk memperkirakan parameter yang diperlukan untuk klasifikasi (Aggarwal, 2013).

Menurut teorema bayes, sebuah probabilitas yang ingin dikomputasi P(C|X) dapat dinyatakan dalam hal probabilitas P(C), P(X|C), dan P(X) dan akan ditunjukkan pada Persamaan 2.2.

$$P(C_k|X_i) = \frac{P(C_k)P(X_i|C_k)}{P(X_i)} \quad (2.2)$$

Keterangan:

$P(C_k|X_i)$ = Peluang kategori *k* ketika terdapat kemunculkan kata *i*

$P(X_i|C_k)$ = Peluang sebuah kata *i* masuk dalam kategori *k*

$P(C_k)$ = Peluang kemunculan sebuah kategori *k*

$P(X_i)$ = Peluang kemunculan sebuah kategori *i*

Pada proses perhitungan klasifikasi peluang kemunculan kata dapat dihilangkan, hal ini dikarenakan peluang tersebut tidak berpengaruh pada perbandingan hasil klasifikasi dari setiap kategori, sehingga proses pada klasifikasi dapat disederhanakan dengan Persamaan 2.3.

$$P(C_k|X_i) = P(C_k)\ P(X_i|C_k) \quad (2.3)$$

Perhitungan *posterior* merupakan perhitungan yang dilakukan dengan mengalikan *prior* dengan *total conditional probability*. Rumus perhitungan *posterior* akan ditunjukkan pada Persamaan 2.4.

$$P(C_k|X_i) = P(C_k) \times P(X_1|C_k) \times P(X_2|C_k) \ldots\ldots P(X_n|C_k) \quad (2.4)$$

2.9.1 Gaussian Naïve Bayes

Gaussian Naïve Bayes adalah algoritme klasifikasi untuk merepresentasikan probabilitas bersyarat dengan menggunakan nilai yang kontinyu. Berikut ini akan ditunjukkan rumus Gaussian Naïve Bayes pada Persamaan 2.5 (John & Langley, 2013).

$$P(x\,|C_k) = \frac{1}{\sqrt{2\pi\sigma^2{}_k}} e^{-\frac{(v-\mu_k)^2}{2\sigma^2{}_k}} \quad (2.5)$$

Keterangan:

x: atribut kontinyu

v: nilai

C_k: kelas

μ_k: rata-rata nilai dalam atribut kontinyu

$\sigma^2{}_k$: variansi nilai dalam atribut kontinyu

2.9.2 Bernoulli Naïve Bayes

Bernoulli Naïve Bayes merepresentasikan dokumen dengan vektor biner yaitu 0 dan 1 yang menggambarkan sebuah masukan. Model ini sangat terkenal dalam pengklasifikasian dokumen yang mana fitur kejadian *term* biner lebih digunakan jika dibandingkan dengan *term frequency* biasa. Berikut ini akan ditunjukkan rumus Bernoulli Naïve Bayes pada Persamaan 2.6 (McCallum & Nigam, 1998).

$$P(x\,|C_k) = \prod_{i=1}^{n} p_{k_i}^{x_i} (1 - p_{ki})^{(1-x_i)} \quad (2.6)$$

Keterangan:

x: atribut kontinyu

C_k: kelas

x_i: sebuah nilai boolean yang mengekspresikan ada atau tidaknya kata dalam sebuah kosa kata

p_{ki}: probabilitas kelas C_k yang menghasilkan sebuah kata

2.9.3 Multinomial Naïve Bayes

Multinomial Naïve Bayes merepresentasikan sebuah dokumen dengan vektor bilangan bulat yang mana unsur-unsurnya menunjukkan *term frequency* yang sesuai dalam dokumen. Berikut ini akan ditunjukkan Rumus Multinomial Naïve Bayes pada Persamaan 2.7 (Lestari, Putra, & Cahyawan, 2010).

$$P(x|C_k) = \frac{1+ N_k}{|V|+N'} \quad (2.7)$$

Parameter $P(x|C_k)$ yang disebut juga *probability likelihood* diestimasikan dengan menghitung kejadian x pada semua dokumen *training* di C_k, menggunakan *Laplacian error* dimana N_k adalah jumlah kemunculan x dalam dokumen pelatihan C_k dan *N'* adalah jumlah total kejadian kata dalam C_k.

2.10 Pengujian

Pengujian merupakan sebuah proses untuk mengkaji sebuah program dengan cara mengumpulkan dan menganalisis informasi tentang aktivitas program, karakteristik, dan hasil program. Tujuan pengujian adalah untuk membuat penilaian, meningkatkan keefektifan, dan menginformasikan hasil keputusan program (Patton, 1987). Pada penelitian ini teknik pengujian yang digunakan adalah *confusion matrix*. *Confusion matrix* merupakan pengujian yang paling optimal untuk klasifikasi *multiclass* yang dalam penelitian ini kelas positif atau negatif (Hossin & Sulaiman, 2015). Berikut ini akan ditunjukkan *confusion matrix* untuk klasifikasi biner pada Tabel 2.7.

Tabel 0.9 *Confusion Matrix*

	Prediksi Kelas Negatif	**Prediksi Kelas Positif**
Aktual Kelas Negatif	*a*	*b*
Aktual Kelas Positif	*c*	*d*

Keterangan:

a = jumlah prediksi yang benar jika sebuah kejadian adalah negatif

b = jumlah prediksi yang salah jika sebuah kejadian adalah positif

c = jumlah prediksi yang salah jika sebuah kejadian adalah negatif

d = jumlah prediksi yang benar jika sebuah kejadian adalah positif

Secara umum beberapa istilah yang didefinisikan untuk dua kelas matriks adalah sebagai berikut:

- *Accuracy (AC)* adalah kesesuaian nilai hasil prediksi pengujian dengan nilai aktual. Rumus *Accuracy* akan ditunjukkan dengan Persamaan 2.8.

$$AC = \frac{a+d}{a+b+c+d} \quad (2.8)$$

- *Recall* atau *True Positive (TP)* adalah jumlah kejadian positif yang diketahui benar. Rumus *Recall* atau *True Positive* akan ditunjukkan dengan Persamaan 2.9.

$$TP = \frac{d}{c+d} \quad (2.9)$$

- *False Positive (FP)* adalah jumlah kejadian negatif yang salah diklasifikasikan sebagai positif. Rumus *False Positive* akan ditunjukkan dengan Persamaan 2.10.

$$FP = \frac{b}{a+b} \quad (2.10)$$

- *True Negative* adalah jumlah kejadian negatif yang diketahui benar. Rumus *True Negative* akan ditunjukkan pada Persamaan 2.11.

$$TN = \frac{a}{a+b} \quad (2.11)$$

- *False Negative* adalah jumlah kejadian positif yang salah diklasifikasikan sebagai negatif. Rumus *False Negative* akan ditunjukkan pada Persamaan 2.12.

$$FN = \frac{c}{c+d} \quad (2.12)$$

- *Precision* adalah jumlah prediksi kejadian positif yang benar. Rumus *Precision* akan ditunjukkan pada Persamaan 2.13.

$$P = \frac{d}{b+d} \quad (2.13)$$

- *F-Measure* adalah pengukuran yang menilai timbal balik antara *precision* dan *recall* (bobot *harmonic mean*). Rumus *F-Measure* akan ditunjukkan pada pada Persamaan 2.14.

$$F = \frac{2PR}{P+R} \quad (2.14)$$

Keterangan:

P: *Precision*

R: *Recall*

BAB 3 METODOLOGI

Pada bab ini, metodologi yang dilakukan dalam penelitian ini akan dijelaskan melalui beberapa subbab yaitu tipe penelitian, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, teknik analisis data, implementasi algoritme dan jadwal penelitian.

3.1 Tipe Penelitian

Pada penelitian ini merupakan tipe penelitian nonimplementatif. Tipe penelitian nonimplementatif memfokuskan investigasi terhadap fenomena atau situasi tertentu, atau analisis terhadap hubungan antar fenomena yang sedang di kaji untuk kemudian menghasilkan hasil investigasi atau hasil analisis ilmiah sebagai produk atau artefak utamanya. Pendekatan dalam penelitian menggunakan pendekatan analitik. Pendekatan analitik merupakan sebuah kegiatan penelitian nonimplementatif yang dilakukan untuk menjelaskan derajat hubungan antar elemen dalam objek penelitian dengan fenomena atau situasi tertentu yang sedang diteliti. Produk atau artefak utamanya yang dihasilkan adalah hasil analisis.

3.2 Strategi Penelitian

Penelitian ini termasuk penelitian yang menggunakan metode eksperimen. Penelitian eksperimen adalah suatu penelitian yang berusaha mencari pengaruh variabel tertentu terhadap variabel yang lain dalam kondisi yang terkontrol secara ketat dan umumnya dilakukan di laboratorium. Metode eksperimen yang digunakan dalam penelitian ini yaitu *ensemble features* dan Naïve Bayes.

3.3 Partisipan Penelitian

Partisipan yang terlibat dalam penelitian ini Guru Bahasa Indonesia SMA Negeri 6 Balikpapan yang bernama Ibu Eka Dewi Lukmana Sari, M.Pd. Alasan pemilihan partisipan penelitian ini agar data yang dikumpulkan dapat menghasilkan data yang valid.

3.4 Lokasi Penelitian

Penelitian ini dilakukan di Laboratorium Komputasi Cerdas Fakultas Ilmu Komputer Universitas Brawijaya. Alasan pemilihan tempat penelitian dilakukan untuk mengukur penggunaan *ensemble features* terhadap hasil analisis sentimen opini film Indonesia pada Twitter.

3.5 Teknik Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data sekunder. Data sekunder adalah sumber data penelitian yang diperoleh peneliti secara tidak langsung melalui media perantara yang mana media perantara yang digunakan dalam penelitian adalah Twitter. Data yang digunakan adalah data *tweet* yang diambil menggunakan Twitter API yang terkait dengan komentar atau *review* film Indonesia yang bersumber dari twitter.com. Data tersebut akan digunakan sebagai data latih dan data uji. Metode yang digunakan dalam pengumpulan data yaitu survei pertanyaan tertutup. Pada pertanyaan tertutup, responden akan mengisi jawaban yang sudah disediakan oleh peneliti. Pengumpulan data mempunyai tujuan untuk mendapatkan data-data yang mempunyai keterkaitan dengan topik dari penelitian. Pengumpulan data dimaksudkan agar mendapatkan bahan-bahan yang relevan, akurat dan dapat diandalkan.

3.6 Teknik Analisis Data

Dalam proses menganalisis data, pengujian merupakan salah satu prosedur penting untuk memastikan program apakah program tersebut dapat berjalan sesuai dengan metode tertentu. Pengujian yang akan dilakukan pada penelitian ini yaitu menggunakan *Confusion Matrix.* Pengujian tersebut bertujuan untuk mengetahui pengaruh fitur *Bag of Words, ensemble features* tanpa fitur *Bag of Words,* dan fitur secara lengkap meliputi fitur *ensemble features* dengan fitur *Bag of Words.*

3.7 Perancangan Algoritme

Proses implementasi algoritme dilakukan sebelum program dikerjakan, pada tahap ini akan dilakukan rancangan secara rinci bagaimana proses pengerjaan dari program dilakukan, agar dapat menjadi acuan saat dikerjakan. Langkah pertama yang dilakukan di dalam implementasi algoritme yaitu memasukkan dokumen hasil sentimen pengguna. Sebagai manualisasi program maka dibutuhkan lima dokumen data latih dan dua dokumen data uji. Kemudian, dokumen tersebut akan dilakukan *text pre-processing*. Untuk menghitung manualisasi data latih maka digunakan *Vector Space Model* dengan *Ensemble Features* yang merupakan fitur gabungan yang meliputi *Twitter Specific features*, *Textual features, Parts of speech (PoS) features*, *Lexicon Based features*, dan *Bag of Words* (BoW) *features*. Setelah dilakukan perhitungan manualisasi data latih, maka dokumen data uji dapat diklasifikasikan hasil sentimennya apakah termasuk kategori positif atau negatif dengan menggunakan *Naïve Bayes Classifier*.

3.8 Jadwal Penelitian

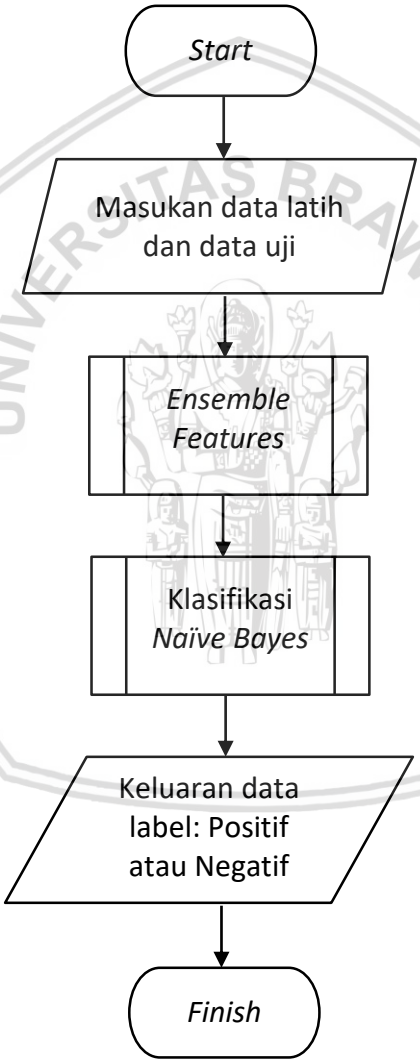
Jadwal penelitian akan dilaksanakan dalam waktu terhitung dari bulan Februari hingga Juni. Berikut ini adalah jadwal penelitian akan ditunjukkan pada Tabel 3.1.

Tabel 0.1 Jadwal Penelitian

No	Uraian	Februari				Maret				April				Mei				Juni			
		Minggu ke-																			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi Kepustakaan																				
2	Pengumpulan Data																				
3	Implementasi Algoritme																				
4	Pengujian dan Analisis																				
5	Kesimpulan dan Saran																				

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Bab ini akan menguraikan tentang tahapan-tahapan yang akan dilakukan dalam membuat analisis sentimen yaitu perancangan diagram alir program, manualisasi, perancangan pengujian, implementasi *ensemble features*, implementasi Naïve Bayes, implementasi pengujian, dan tampilan program sistem. Tahapan-tahapan utama program dimulai dari *start,* masukan data latih dan data uji, *ensemble features*, klasifikasi Naïve Bayes, keluaran data latih dan data uji, dan *return*. Berikut ini adalah tahapan-tahapan utama program yang akan ditunjukkan pada Gambar 4.1.



Gambar 4.1 Tahapan-tahapan Utama pada Program

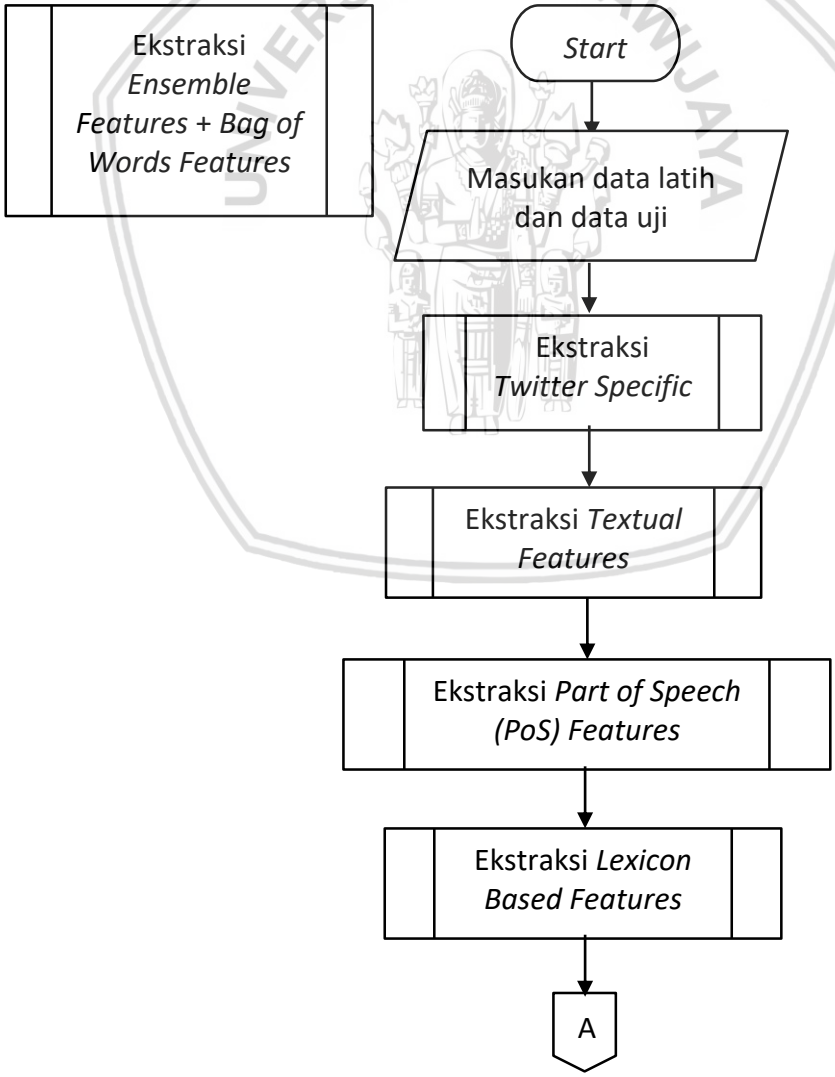
4.1 Tahapan-tahapan Utama Program

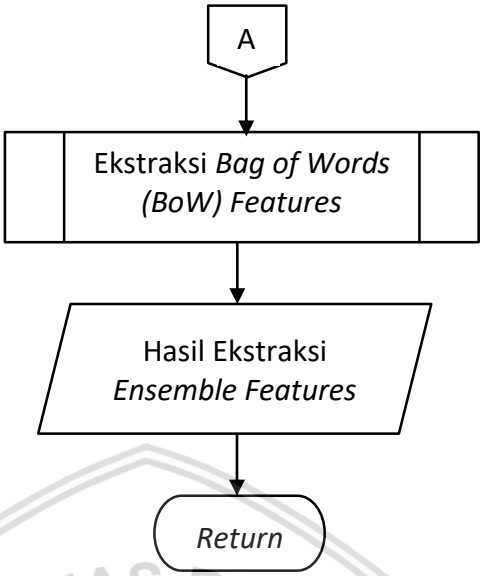
Tahapan-tahapan utama program pada penelitian ini merupakan alur kinerja yang dilakukan untuk menyelesaikan sebuah permasalahan. Tahapan awal dalam

penelitian dimulai dari memasukkan data latih dan data uji kemudian dilakukan perhitungan nilai fitur dengan menggunakan *ensemble features* yaitu *twitter specific features, textual features, Part of Speech (PoS) features, lexicon based features,* dan *Bag of Words (BoW) features.* Pada *Bag of Words* akan dilakukan *preprocessing text* terlebih dahulu yang terdiri dari *case folding*, *data cleaning*, *stemming*, *filtering*, dan tokenisasi. Kemudian, dilakukan pengklasifikasian menggunakan Naïve Bayes untuk mendapatkan data label apakah termasuk kategori positif atau negatif. Alur kinerja dari tahapan-tahapan di atas akan dijelaskan subbab-subbab berikut ini.

4.1.1 Tahapan pada Ensemble Features

Pada tahapan ini akan dilakukan ekstraksi fitur dengan menggunakan *ensemble features* dengan menggunakan sebanyak 37 fitur. Pada tahap ini terdapat 5 tahapan utama yaitu *Twitter specific features, textual features, Part of Speech (PoS) features, lexicon based features,* dan *Bag of Words (BoW).* Berikut ini adalah tahapan-tahapan pada *ensemble features* yang akan ditunjukkan pada Gambar 4.2.

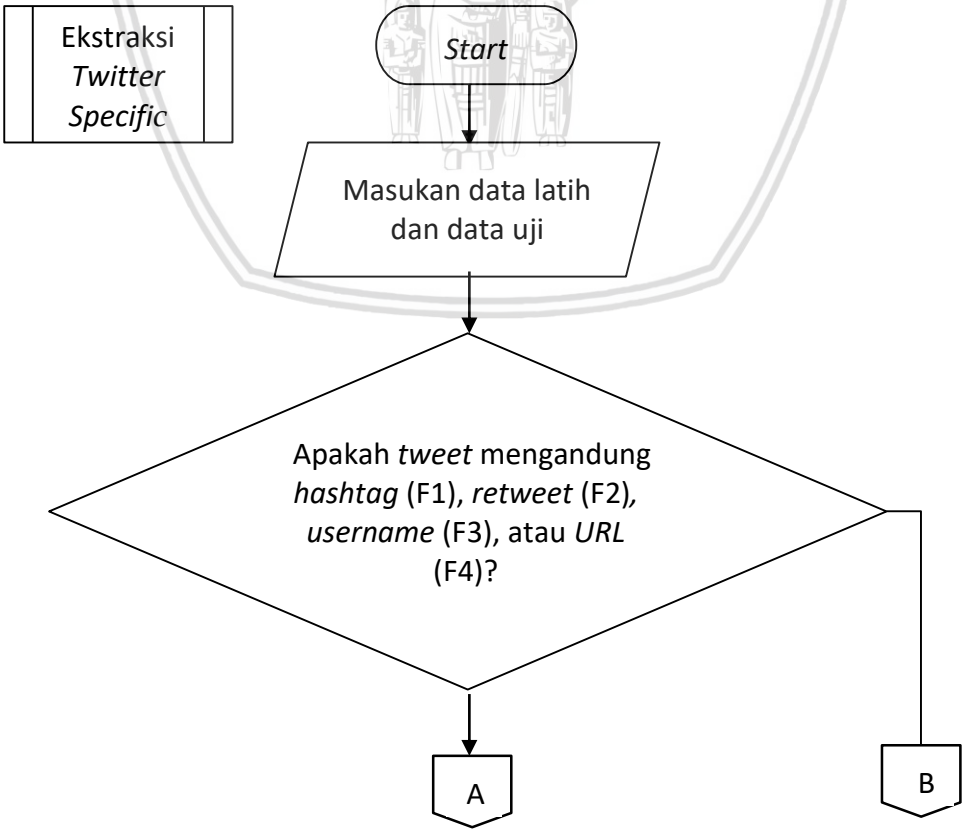


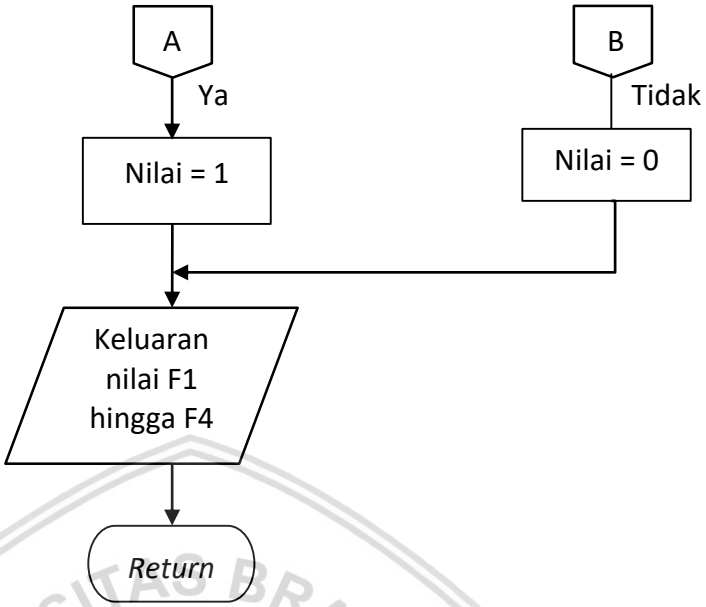


Gambar 4.2 Tahapan pada *Ensemble Features*

4.1.1.1 *Twitter Specific*

Pada bagian ini, proses perhitungan nilai fitur dilakukan dengan melihat pada spesifikasi yang terdapat pada sebuah *tweet.* Pada tipe fitur ini terdapat 4 fitur yang terdiri dari F1 hingga F4 yaitu apakah *tweet* mengandung *hashtag* (F1), *retweet* (F2), *username* (F3), atau *URL* (F4). Berikut ini adalah tahapan pada *Twitter specific* yang akan ditunjukkan pada Gambar 4.3.

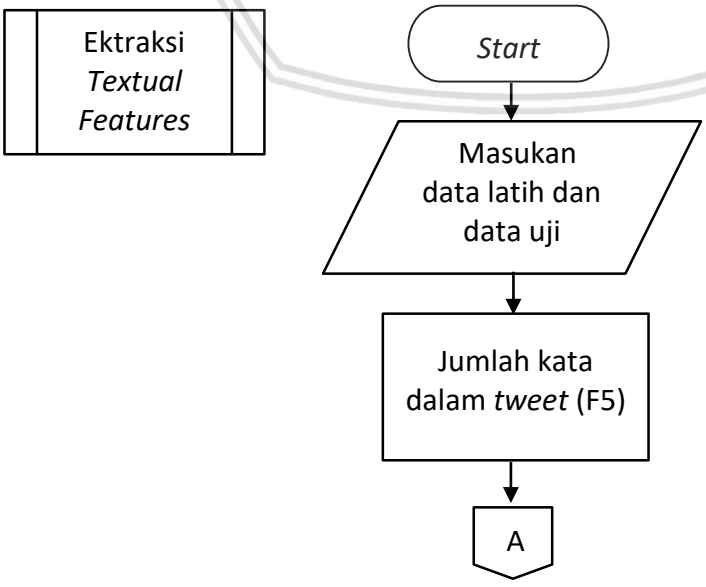


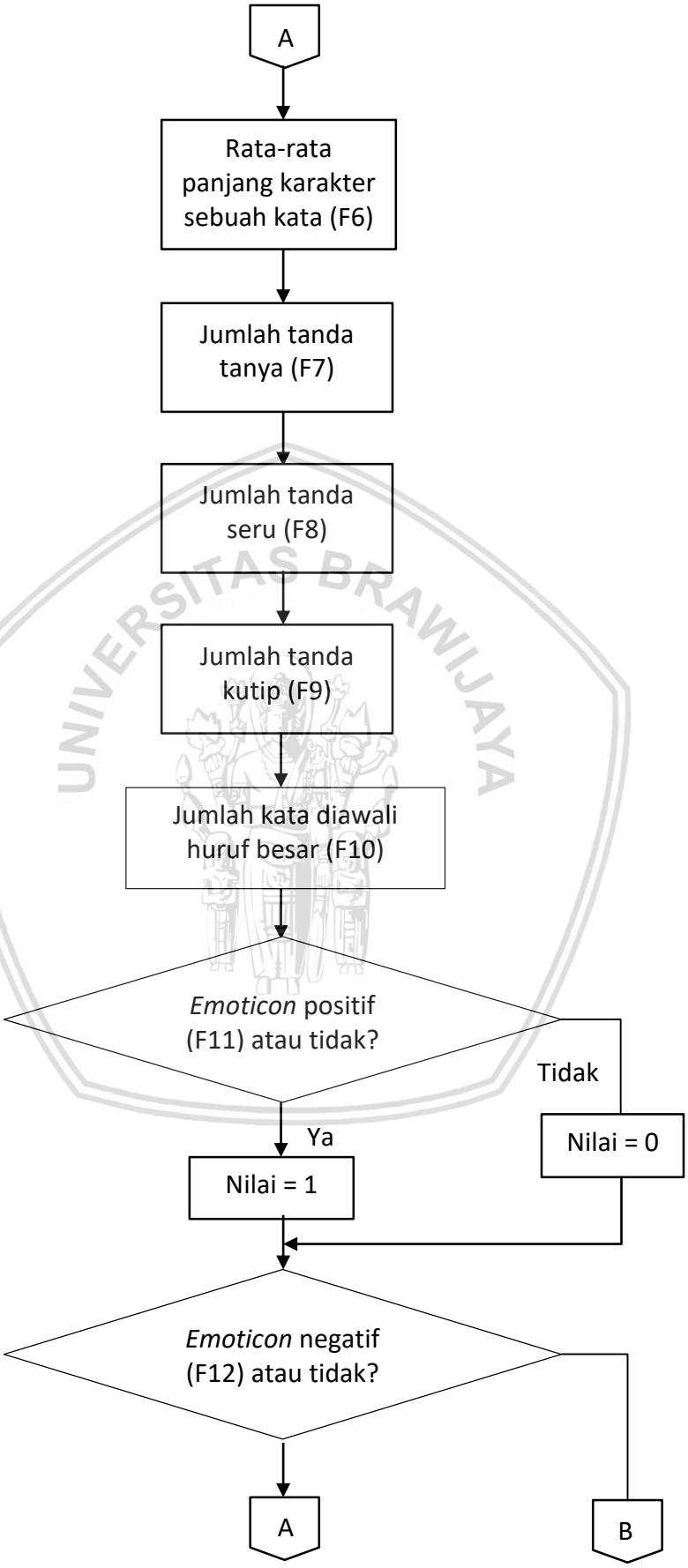


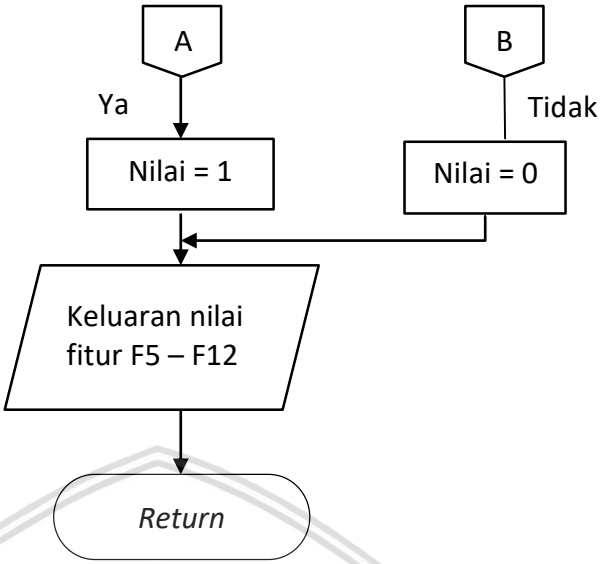
Gambar 4.3 Tahapan pada *Twitter Specific*

4.1.1.2 *Textual Features*

Pada bagian ini, proses perhitungan nilai fitur dilakukan dengan melihat pada informasi eksplisit yang ada pada sebuah *tweet*. Pada tipe fitur ini terdapat 8 fitur yang terdiri dari F5-F12 yaitu jumlah kata dalam *tweet* (F5), rata-rata panjang karakter pada sebuah kata (F6), jumlah tanda tanya (F7), jumlah tanda seru (F8), jumlah tanda kutip (F9), jumlah kata yang diawali huruf besar (F10), apakah termasuk *emoticon* positif (F11) atau negatif (F12). Berikut ini adalah tahapan *textual features* akan ditunjukkan pada Gambar 4.4.



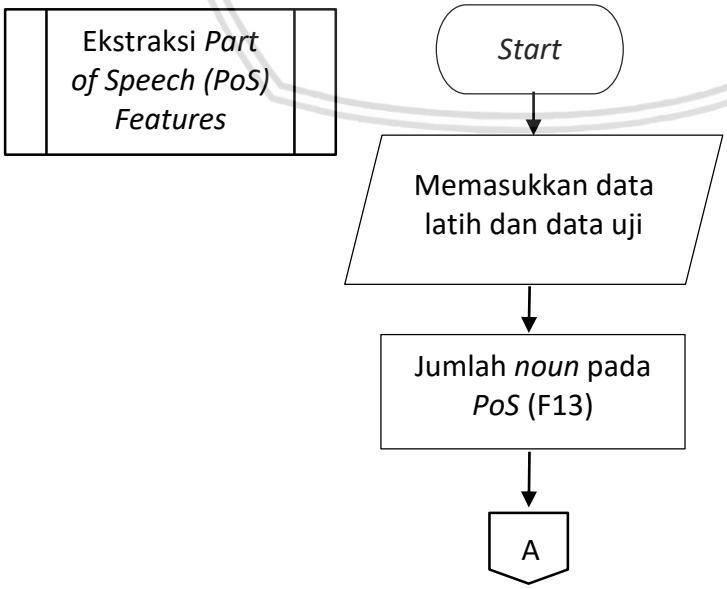


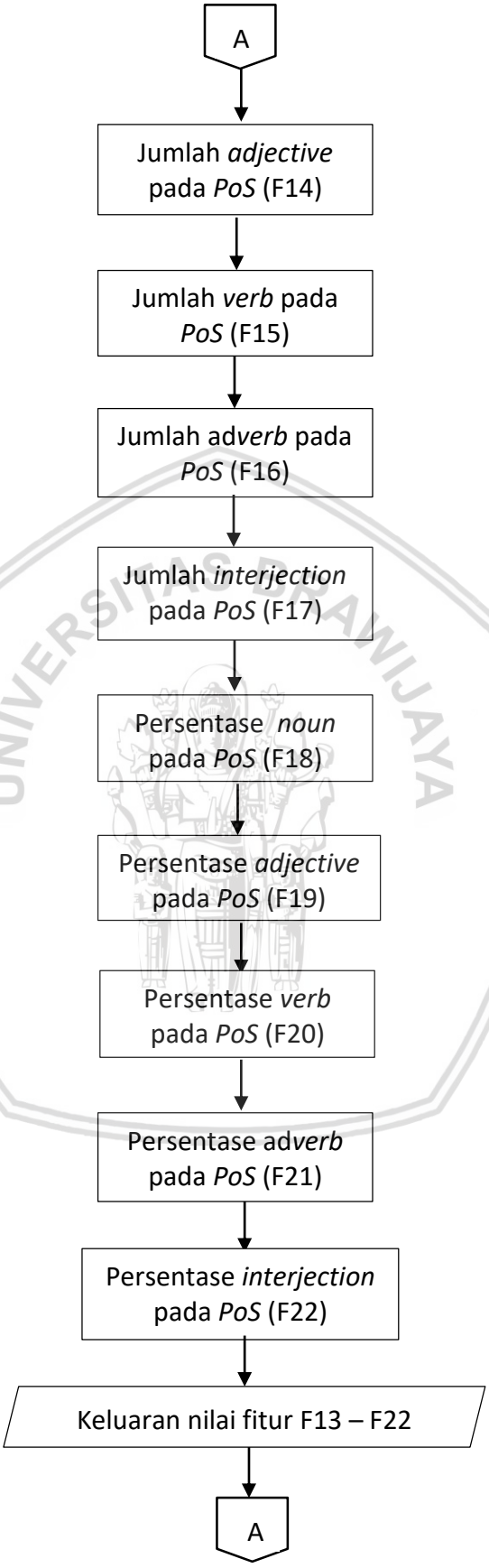


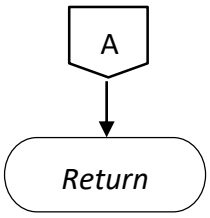
Gambar 4.4 Tahapan pada *Textual Features*

4.1.1.3 *Part of Speech (PoS) Features*

Pada bagian ini, proses perhitungan nilai fitur dilakukan dengan melihat pada informasi kata yang ada pada sebuah *tweet*. Dengan memanfaatkan *lexicon* maka *Part of Speech features* dapat diekstraksi. Pada tipe fitur ini terdapat 10 fitur, yaitu jumlah *noun* pada *PoS* (F13), jumlah *adjective* pada *PoS* (F14), jumlah *verb* pada *PoS* (F15), jumlah *adverb* pada *PoS* (F16), jumlah *interjection* pada *PoS* (F17), persentase *noun* pada *PoS* (F18), persentase *adjective* pada *PoS* (F19), persentase *verb* pada *PoS* (F20), persentase *adverb* pada *PoS* (F21), dan persentase *interjection* pada *PoS* (F22). Berikut ini adalah tahapan pada *Part of Speech (PoS) Features* akan ditunjukkan pada Gambar 4.5.



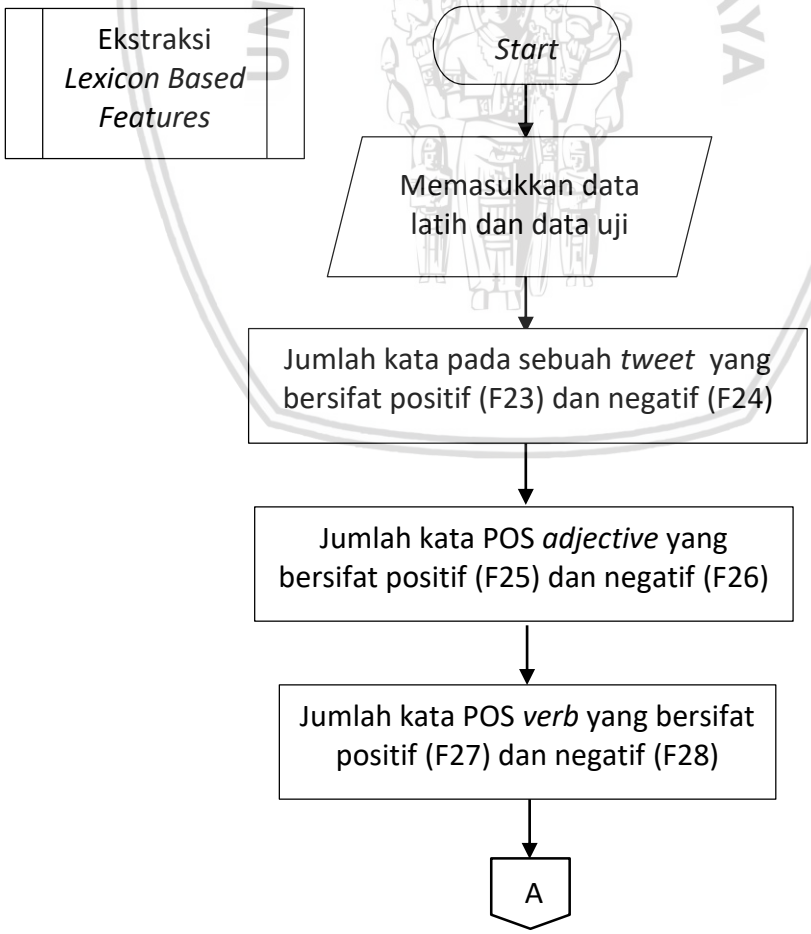

A
Jumlah *adjective* pada *PoS* (F14)
Jumlah *verb* pada *PoS* (F15)
Jumlah ad*verb* pada *PoS* (F16)
Jumlah *interjection* pada *PoS* (F17)
Persentase *noun* pada *PoS* (F18)
Persentase *adjective* pada *PoS* (F19)
Persentase *verb* pada *PoS* (F20)
Persentase ad*verb* pada *PoS* (F21)
Persentase *interjection* pada *PoS* (F22)
Keluaran nilai fitur F13 – F22
A

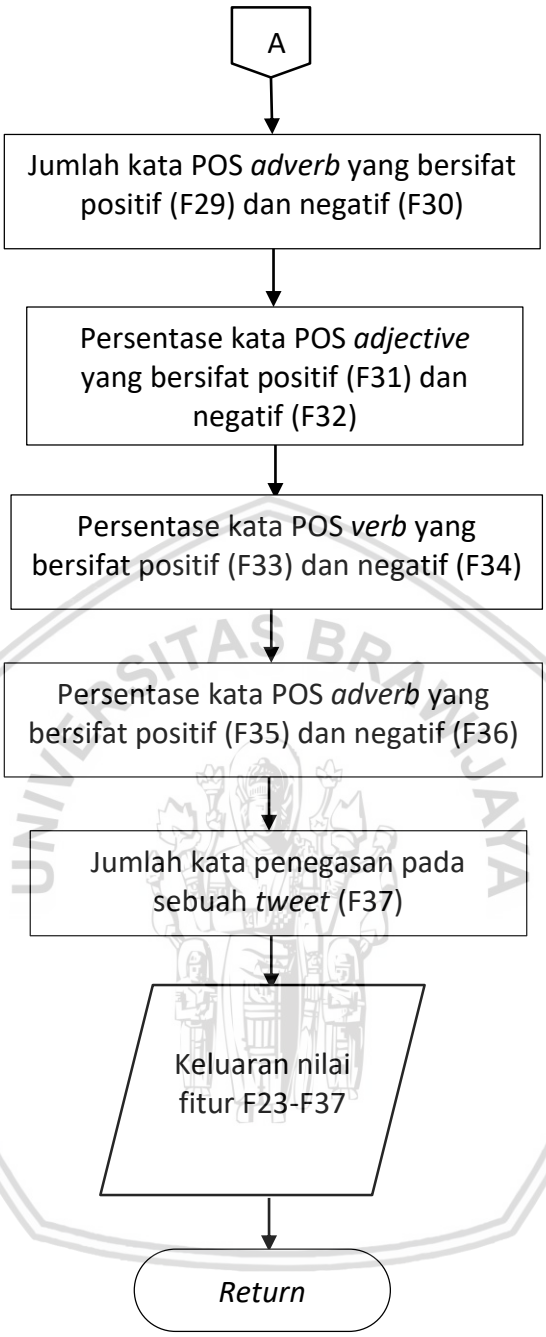


Gambar 4.5 Tahapan pada *Part of Speech Features*

4.1.1.4 *Lexicon Based Features*

Pada bagian ini, proses pengekstrasian nilai fitur dilakukan dengan memanfaatkan *lexicon* atau kamus kata. Pada tipe fitur ini terdapat 15 fitur,yang terdiri dari jumlah kata positif pada sebuah *tweet* (F23), jumlah kata negatif pada sebuah *tweet* (F24), jumlah kata POS *adjective* yang bersifat positif (F25) dan negatif (F26), jumlah kata POS *verb* yang bersifat positif (F27) dan negatif (F28), jumlah kata POS *adverb* yang bersifatpositif (F29) dan negatif (F30), persentase kata POS *adjective* yang bersifat positif (F31) dan negatif (F32), persentase kata POS *verb* yang bersifat positif (F33) dan negatif (F34), persentase kata POS *adverb* yang bersifat positif (F35), dan negatif (F36), dan jumlah kata penegasan (F37). Berikut ini adalah tahapan pada *Lexicon Based Features* yang akan ditunjukkan pada Gambar 4.6.

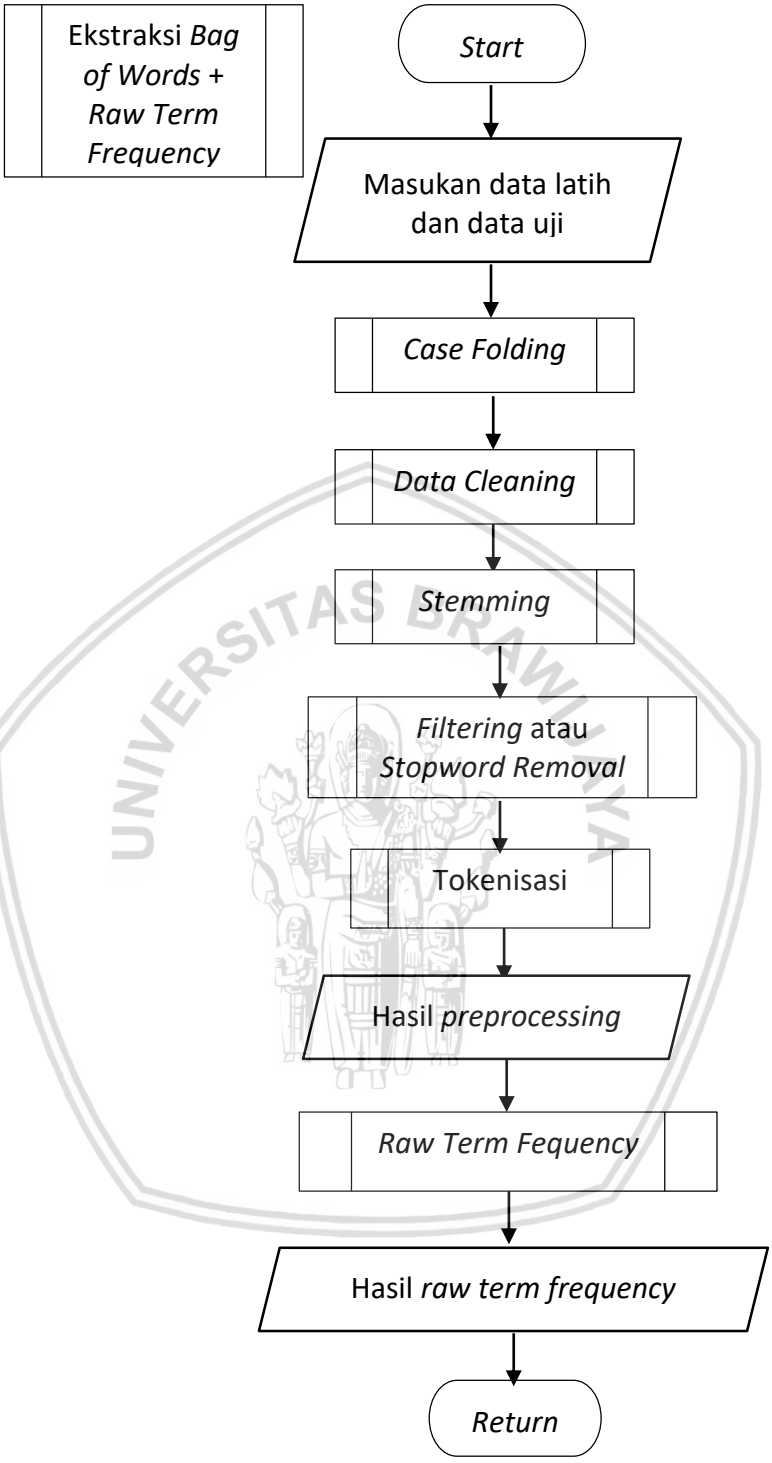




Gambar 4.6 Tahapan pada *Lexicon Based Features*

4.1.2 Tahapan pada *Bag of Word Features* dan *Raw Term Frequency*

Pada tahapan *bag of words features*, cara pengektrasian nilai fitur dengan tidak memperhatikan urutan dari sebuah kalimat. Untuk mendapatkan nilai fitur pada tahapan ini dilakukan dengan cara melakukan *text preprocessing* terlebih dahulu selanjutnya akan dilakukan *Raw Term Frequency* untuk mendapatkan bobot kata. Berikut ini adalah tahapan pada *Bag of WordsFeatures* yang akan ditunjukkan pada Gambar 4.7.

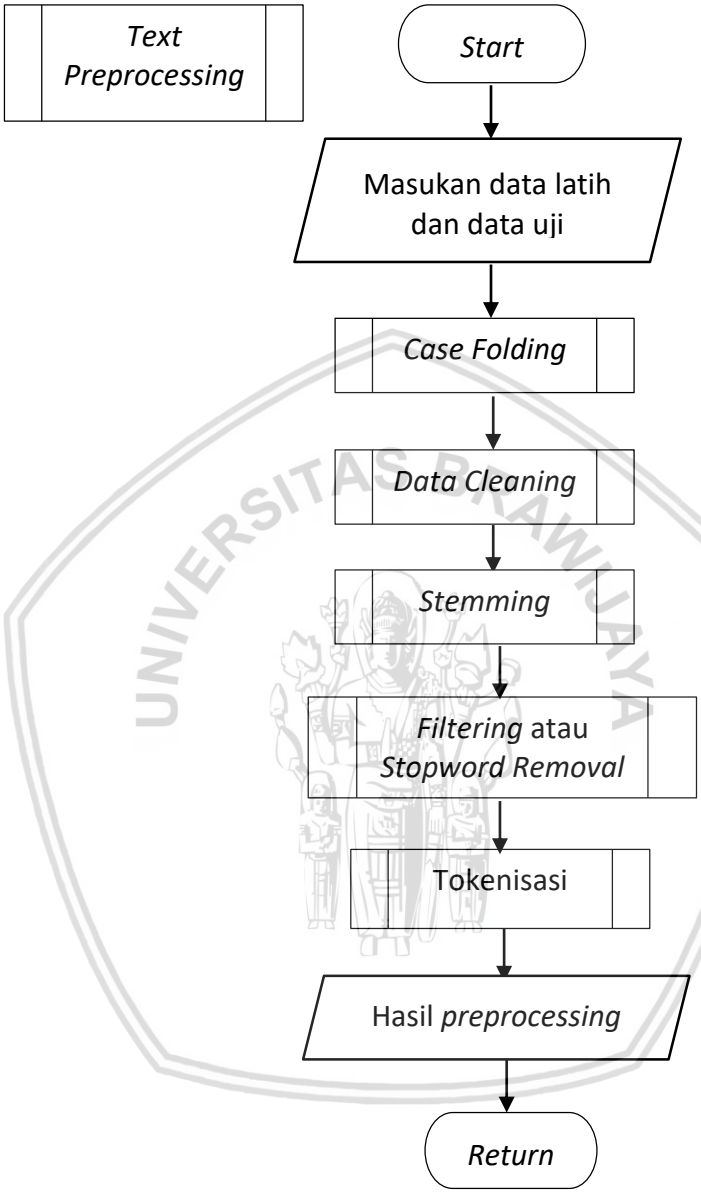


Gambar 4.7 Tahapan pada *Bag of Words Features*

4.1.3 *Text Preprocessing*

Text preprocessing merupakan proses pembersihan dan penyiapan teks untuk klasifikasi. *Text preprocessing* merupakan bagian penting dari pemrosesan bahasa alami di mana karakter, kata, dan kalimat yang diidentifikasi pada tahap

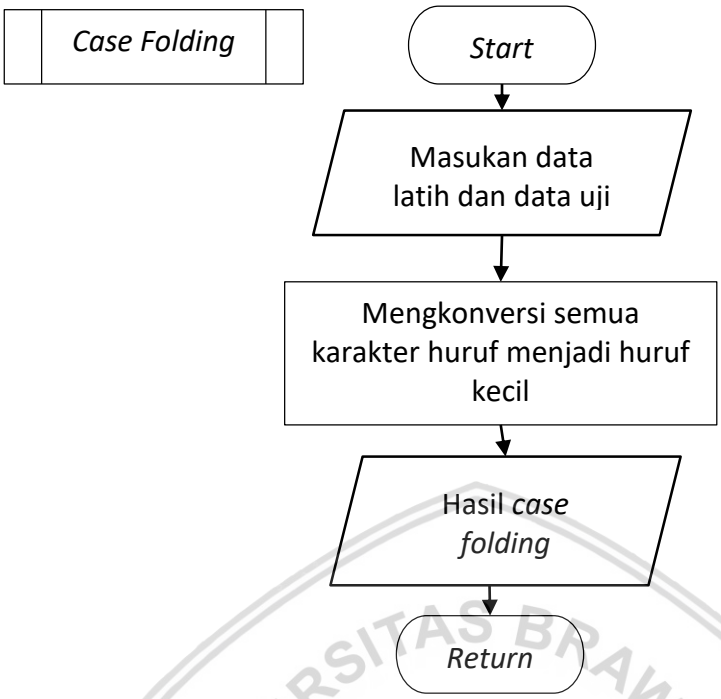
ini adalah unit yang mendasar yang nantinya akan diproses pada tahap selanjutnya. Berikut ini adalah tahapan *text preprocessing* yang akan ditunjukkan pada Gambar 4.8.



Gambar 4.8 Tahapan pada *Text Preprocessing*

4.1.3.1 *Case Folding*

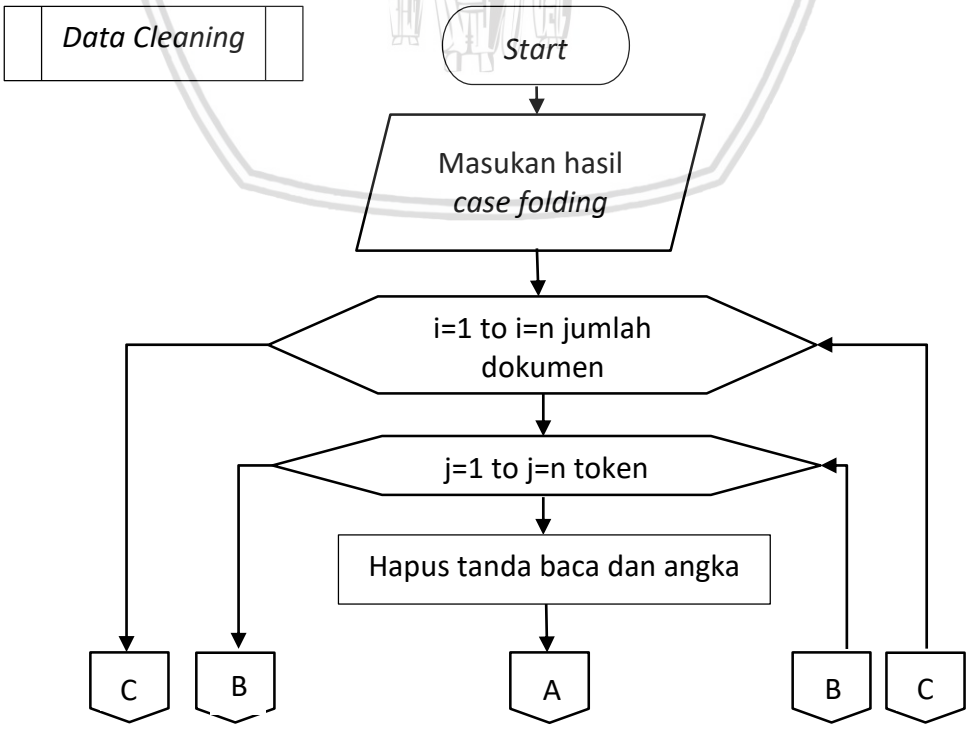
Tahapan *case folding* adalah tahapan untuk mengonversi sebuah dokumen menjadi huruf kecil semua atau *lowercase*. Tahapan ini akan dilakukan apabila dokumen memiliki kata yang mengandung huruf besar. Berikut ini adalah tahapan *case folding* yang akan ditunjukkan pada Gambar 4.9.

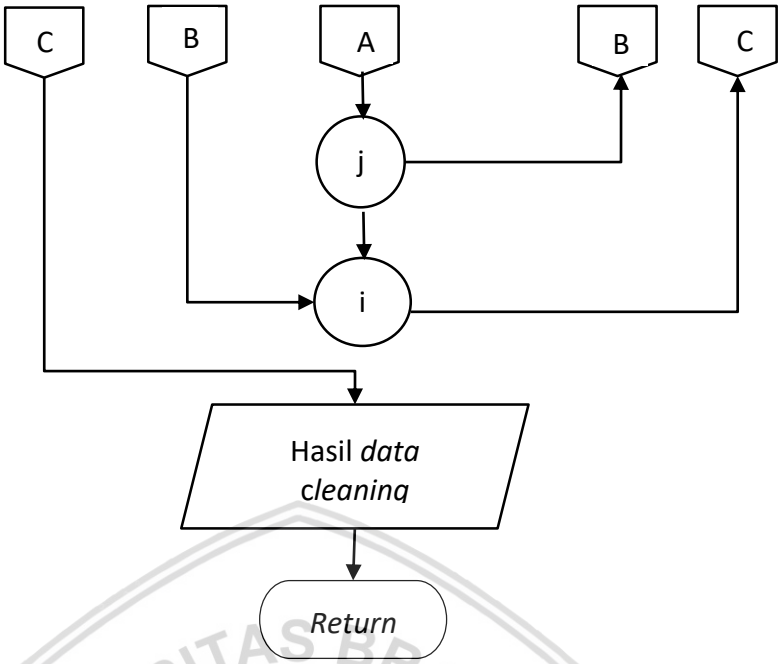


Gambar 4.9 Tahapan pada *Case Folding*

4.1.3.2 *Data Cleaning*

Tahapan *data cleaning* adalah tahapan untuk menghilangkan karakter yang tidak penting pada dokumen teks. Karakter tidak penting yang akan dihilangkan sebagai contoh tanda baca dan angka. Berikut ini adalah tahapan *cleaning* yang akan ditunjukkan pada Gambar 4.10.

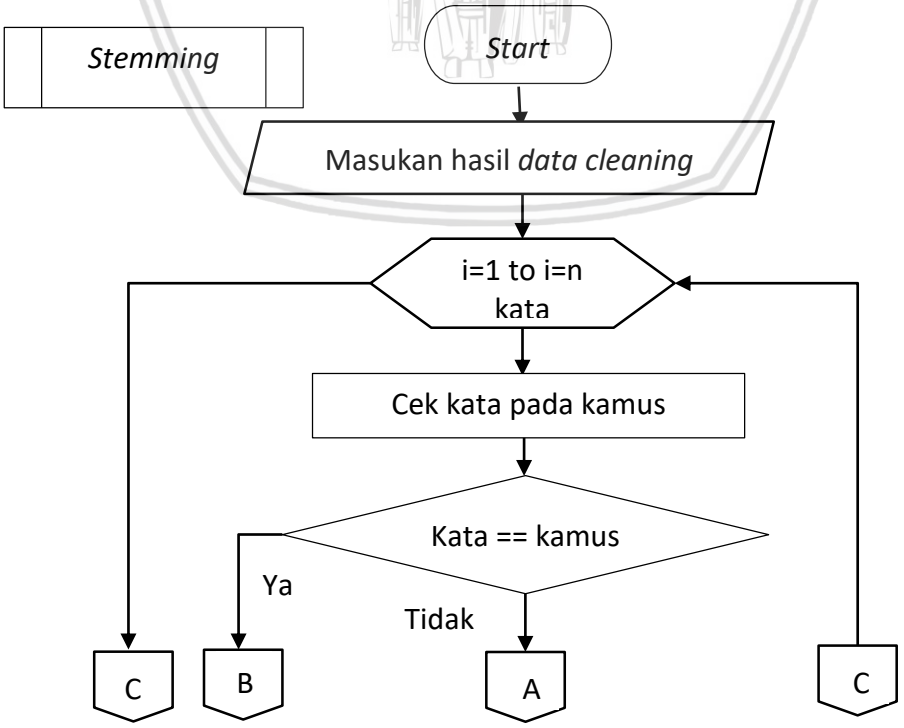


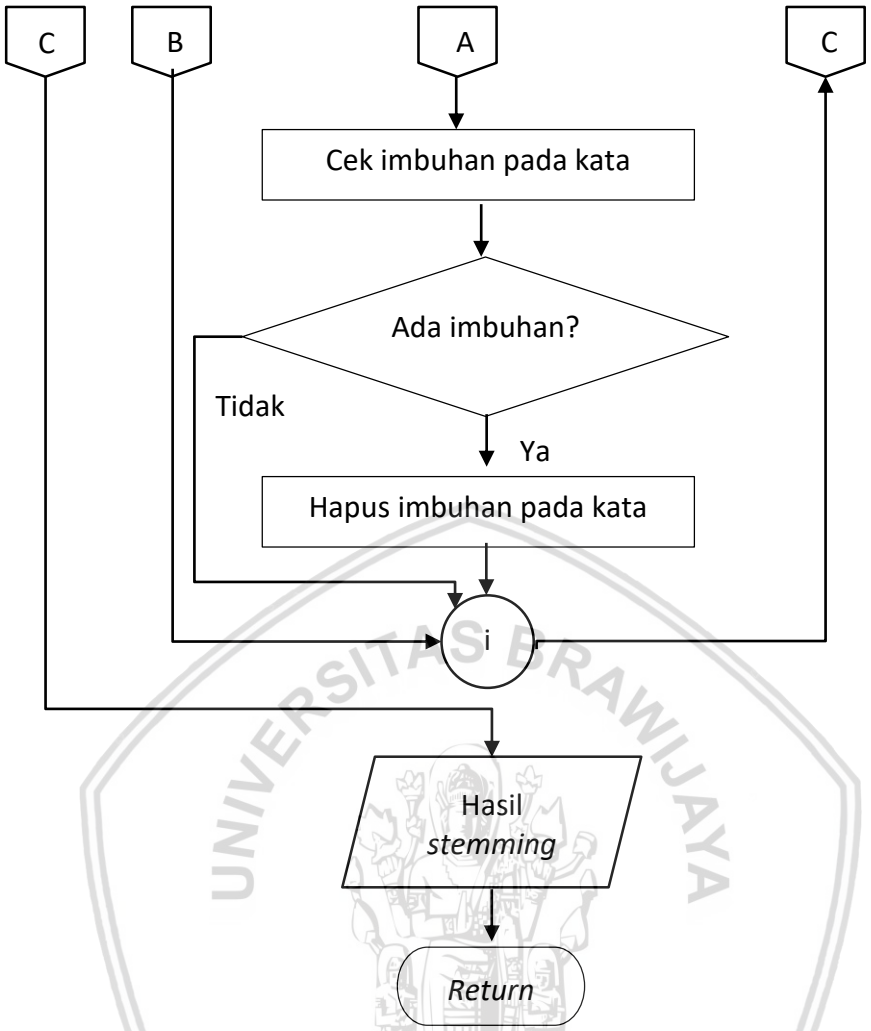


Gambar 4.10 Tahapan pada *Data Cleaning*

4.1.3.3 *Stemming*

Tahapan *Stemming* adalah tahap mengubah atau menguraikan bentuk kata menjadi kata dasar atau tahapan mencari dasar kata dari tiap kata hasil dari *filtering* atau *stopword removal*. Proses *stemming* akan dilakukan setelah proses *filtering* dilakukan. Berikut ini adalah tahapan *stemming* yang akan ditunjukkan pada Gambar 4.11.





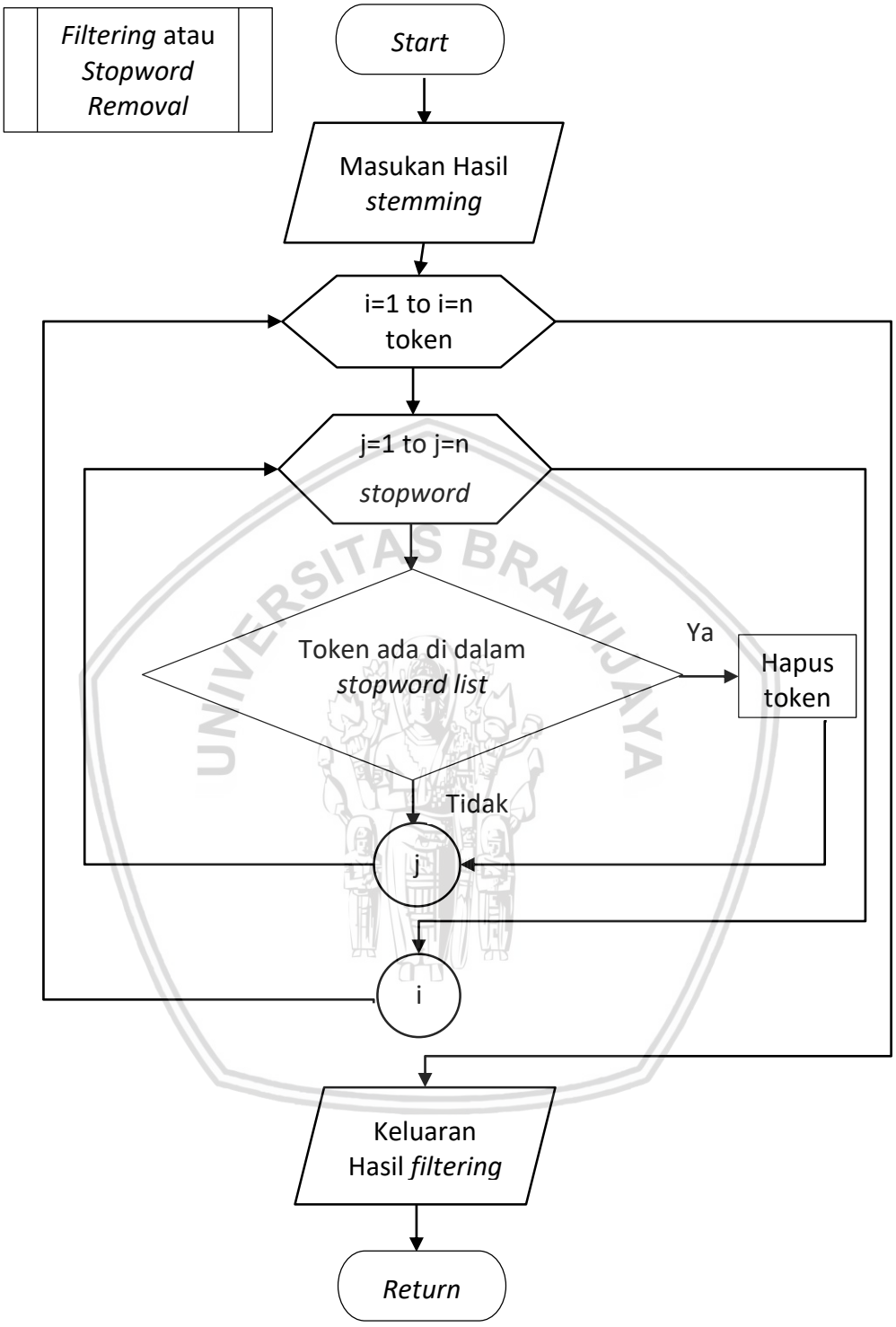
Gambar 4.11 Tahapan pada *Stemming*

4.1.3.4 *Filtering* atau *Stopword Removal*

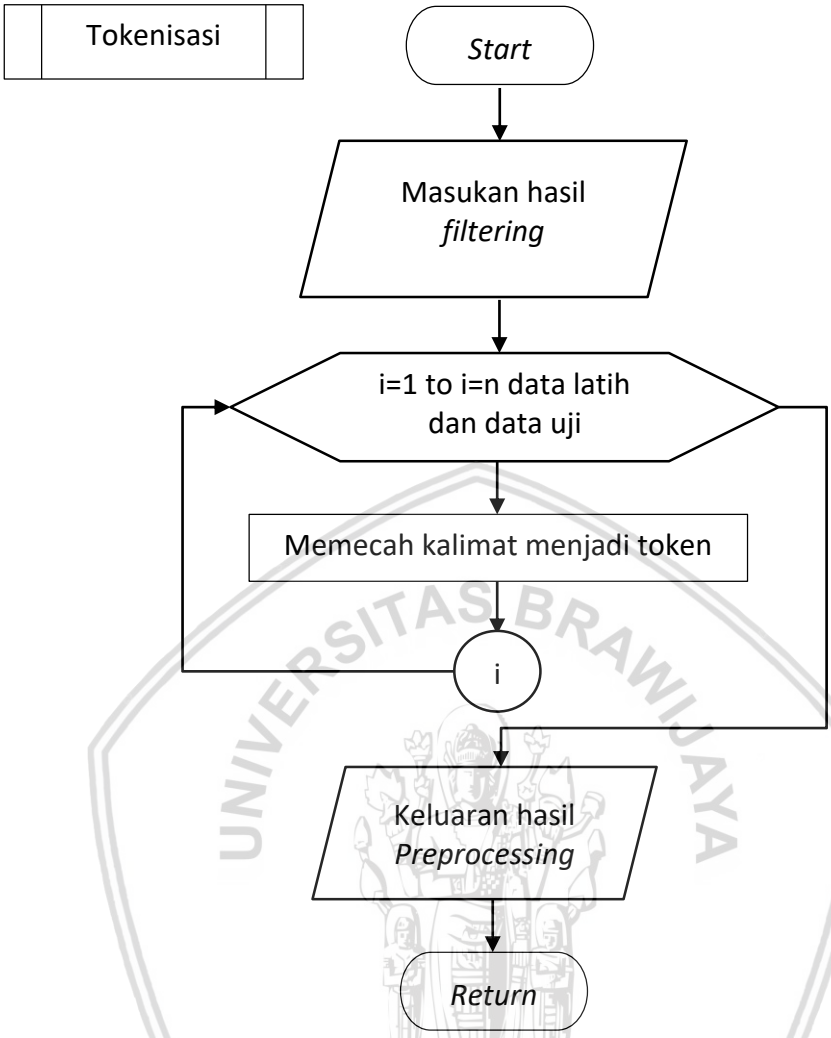
Tahapan *Filtering* atau *Stopword Removal* adalah tahap pemilihan kata-kata penting dari hasil token, yaitu kata-kata apa saja yang akan digunakan untuk mewakili dokumen. Pemilihan kata-kata tersebut didapatkan dari kamus *stopword list*. Berikut ini adalah tahapan *filtering* yang akan ditunjukkan pada Gambar 4.12.

4.1.3.5 Tokenisasi

Tahapan Tokenisasi merupakan tahap awal pada saat melakukan *text preprocessing*. Tahapan Tokenisasi adalah proses pemecahan aliran teks ke dalam kata-kata, ungkapan, simbol, atau elemen bermakna lainnya yang disebut token. Hasil token tersebut akan diproses pada tahap selanjutnya. Berikut ini adalah tahapan tokenisasi yang akan ditunjukkan pada Gambar 4.13.



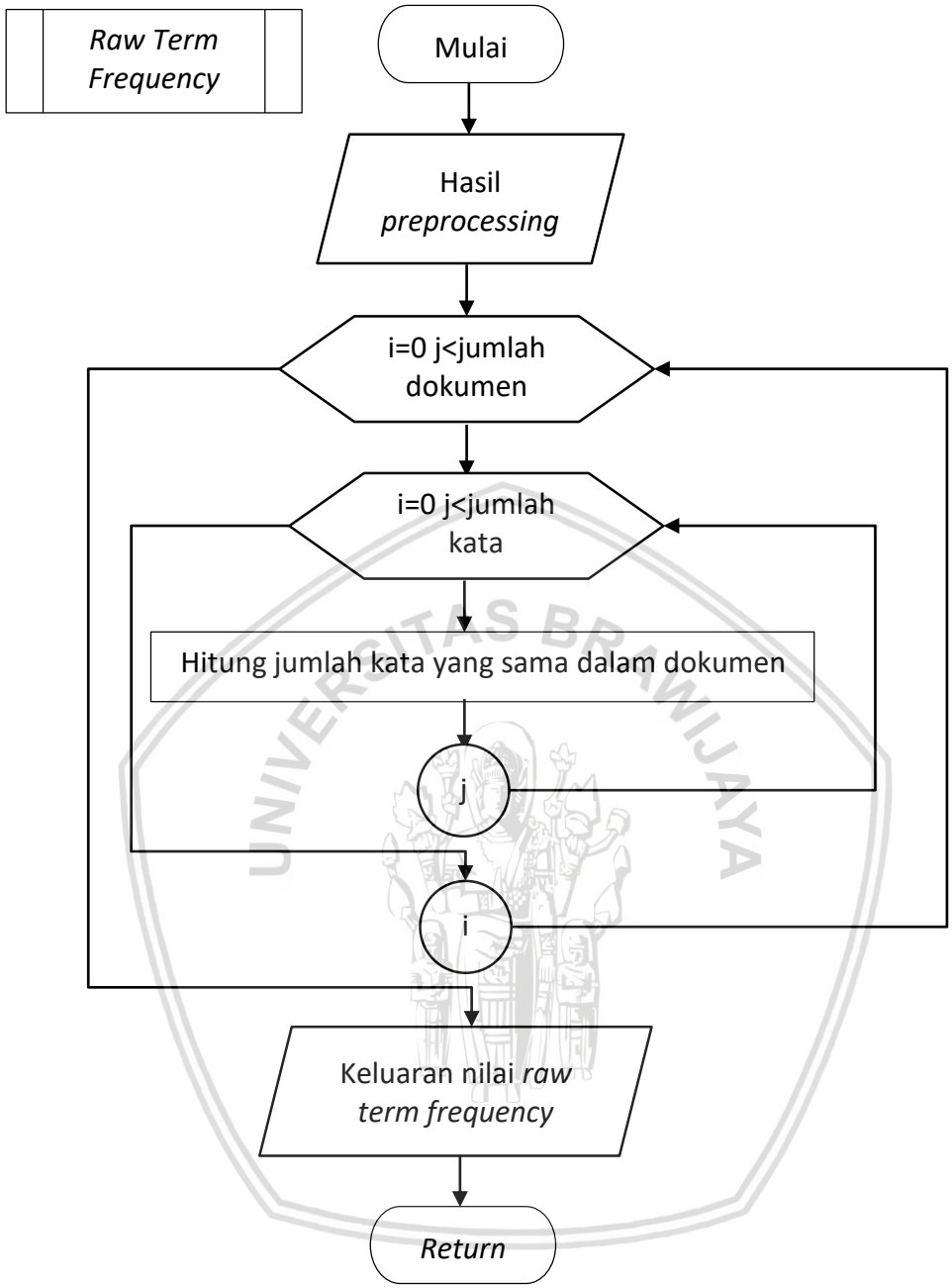
Gambar 4.12 Tahapan pada *Filtering* atau *Stopword Removal*



Gambar 4.13 Tahapan pada Tokenisasi

4.1.3.6 *Raw Term Frequency*

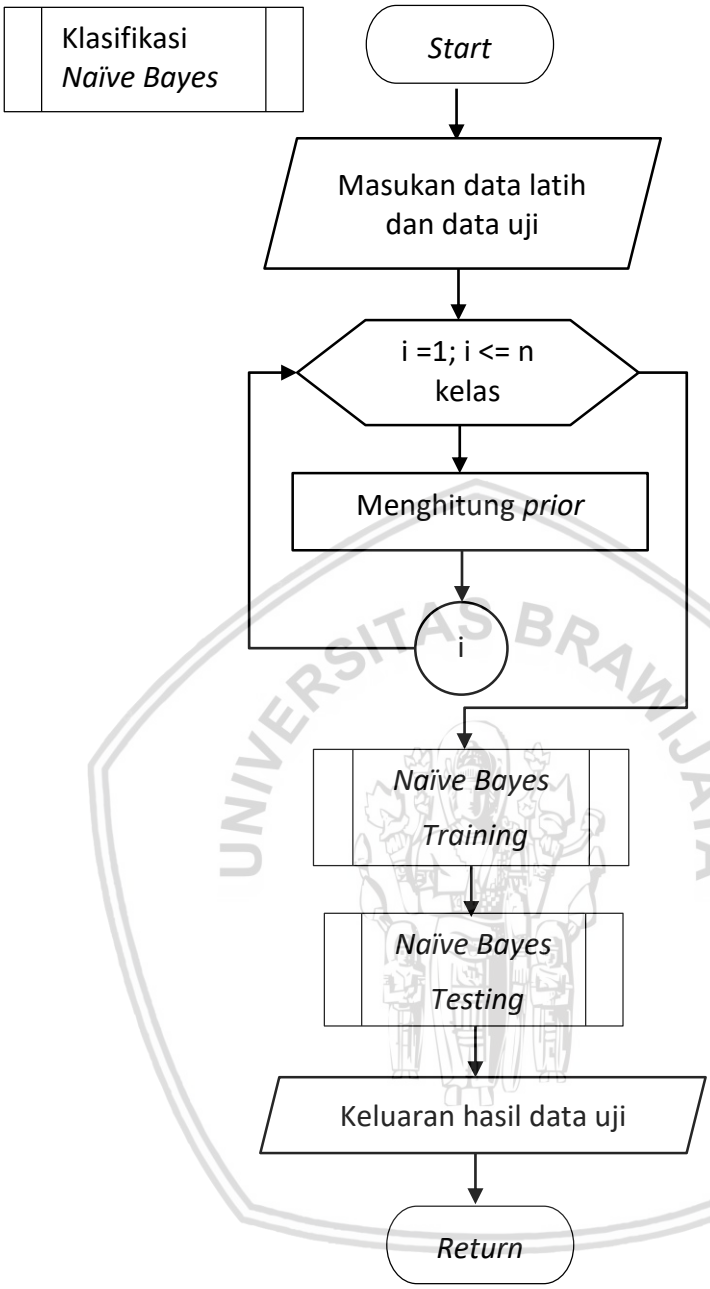
Raw Term Frequency menganggap semua *term* dianggap sama pentingnya dalam hal menilai relevansi pada kueri. Sebenarnya beberapa *term* tertentu hanya sedikit atau bahkan tidak memiliki kekuatan dalam menentukan relevansinya (Manning et al., 2008). Berikut ini adalah proses *raw term frequency* yang akan ditunjukkan pada Gambar 4.14.



Gambar 4.14 Proses *Raw Term Frequency*

4.1.4 Klasifikasi *Naïve Bayes*

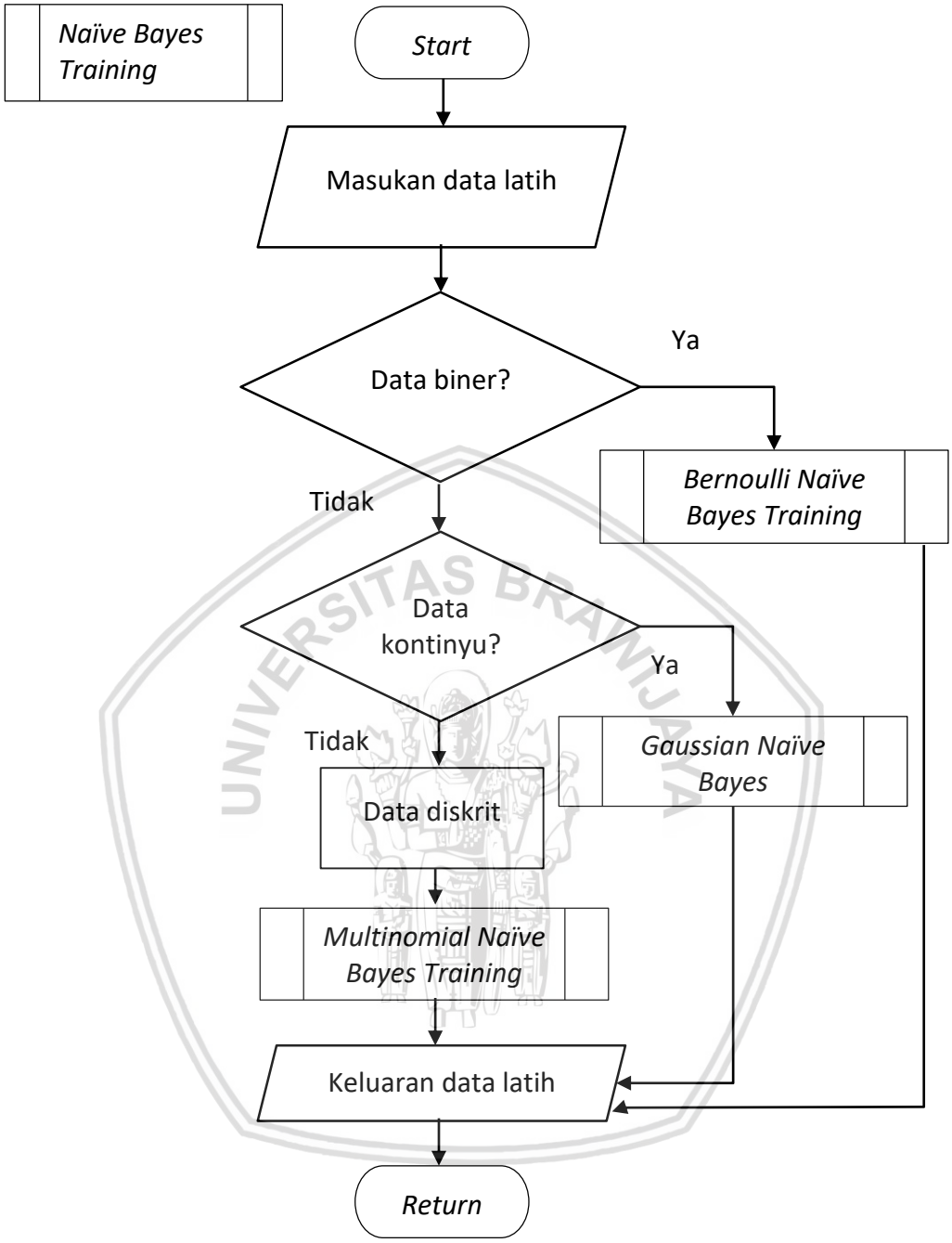
Dalam menyelesaikan permasalahan, metode Naïve Bayes yang digunakan adalah Multinomial Naïve Bayes, Gaussian Naïve Bayes, dan Bernoulli Naïve Bayes. Distribusi Multinomial Naïve Bayes untuk menyelesaikan permasalahan dengan data diskrit, Gaussian Naïve Bayes untuk menyelesaikan permasalahan dengan data kontinyu, dan Bernoulli Naïve Bayes untuk menyelesaikan permasalahan dengan data biner. Berikut ini adalah tahapan klasifikasi Naïve Bayes yang akan ditunjukkan pada Gambar 4.15.



Gambar 4.15 Tahapan pada Klasifikasi *Naïve Bayes*

4.1.4.1 *Naïve Bayes Training*

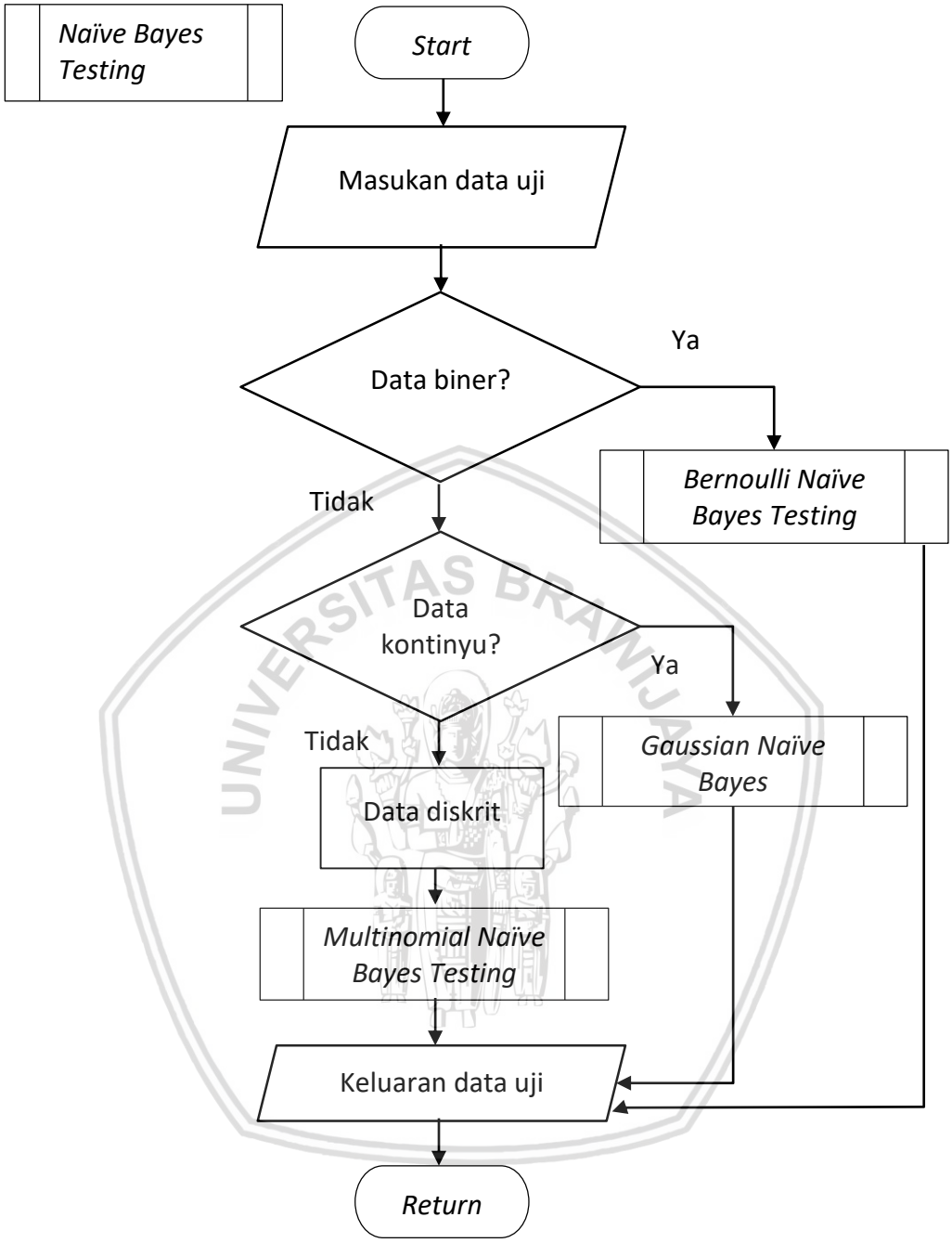
Pada tahapan ini akan dijelaskan diagram alir tentang Naïve Bayes *Training* meliputi Bernoulli Naïve Bayes *Training,* Multinomial Naïve Bayes *Training,* dan Gaussian Naïve Bayes. Berikut ini adalah tahapan Naïve Bayes *Training* yang akan ditunjukkan pada Gambar 4.16.



Gambar 4.16 Tahapan pada Naïve Bayes *Training*

4.1.4.2 *Naïve Bayes Testing*

Pada tahapan ini akan dijelaskan diagram alir tentang Naïve Bayes *Testing* meliputi Bernoulli Naïve Bayes *Testing,* Multinomial Naïve Bayes *Testing,* dan Gaussian Naïve Bayes. Berikut ini adalah tahapan Naïve Bayes *Training* yang akan ditunjukkan pada Gambar 4.17.



Gambar 4.17 Tahapan pada Naïve Bayes *Testing*

4.1.4.3 *Bernoulli Naïve Bayes Training*

Dalam tahapan distribusi *Bernoulli Naïve Bayes Training*, proses perhitungan melihat pada representasi data biner apakah data tersebut bernilai 1 (terdapat fitur) atau 0 (tidak terdapat fitur). Berikut ini adalah tahapan *Bernoulli Naïve Bayes* pada data latih yang akan ditunjukkan pada Gambar 4.18.

4.1.4.4 *Bernoulli Naïve Bayes Testing*

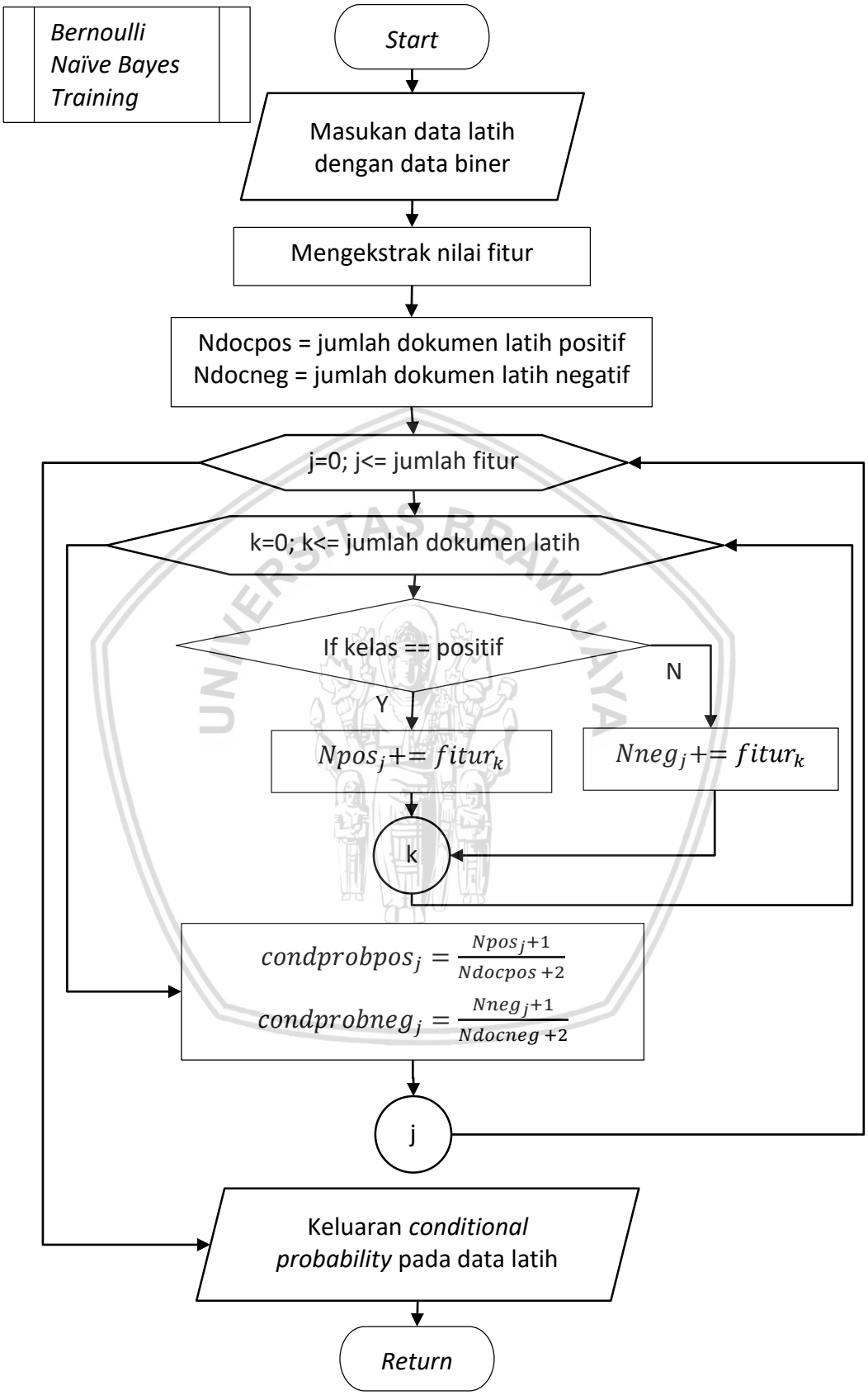
Dalam tahapan distribusi *Bernoulli Naïve Bayes Testing*, proses perhitungan melihat pada representasi data biner apakah data tersebut bernilai 1 (terdapat fitur) atau 0 (tidak terdapat fitur). Berikut ini adalah tahapan *Bernoulli Naïve Bayes* pada data uji yang akan ditunjukkan pada Gambar 4.19.

4.1.4.5 *Gaussian Naïve Bayes*

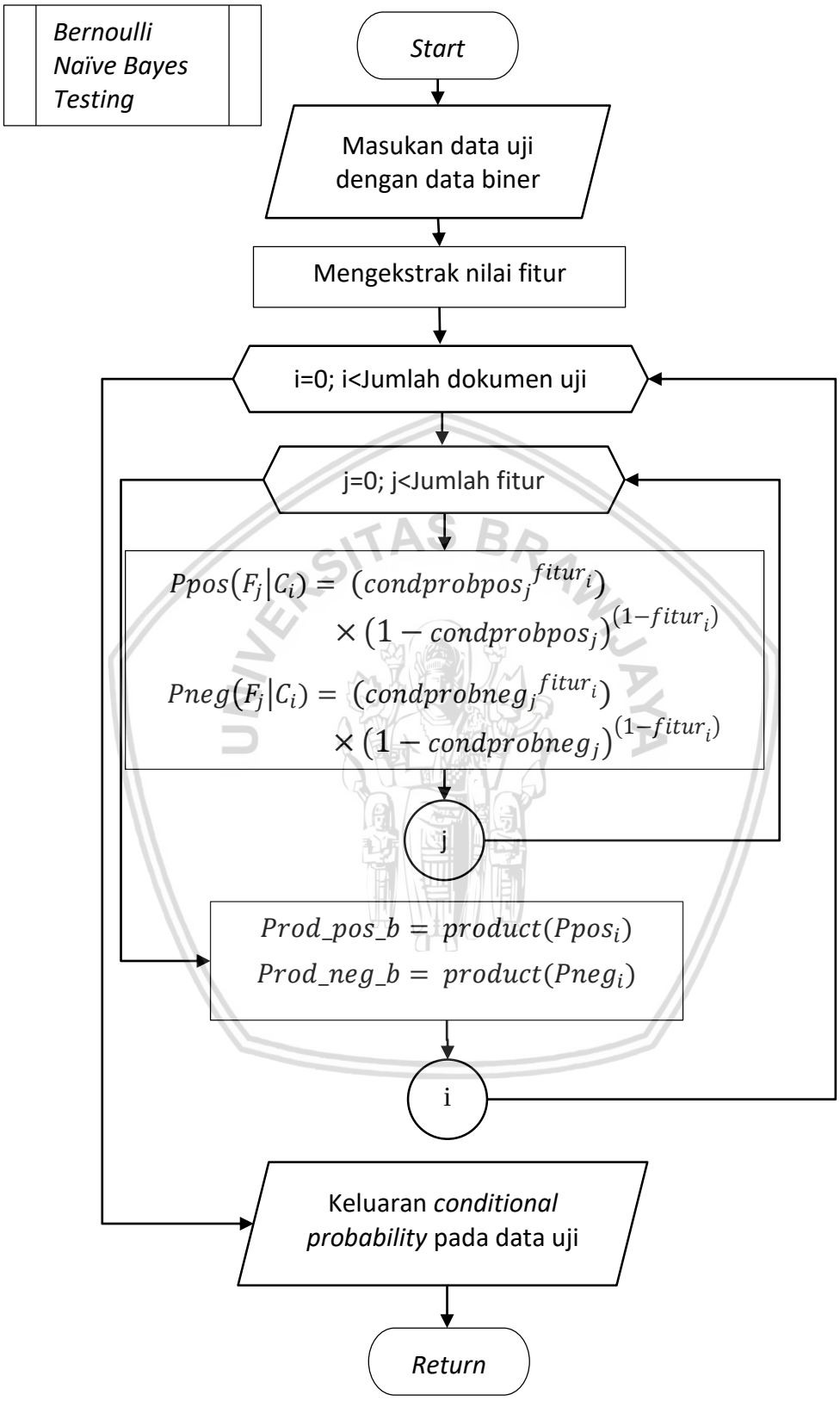
Dalam tahapan distribusi *Gaussian Naïve Bayes*, proses perhitungan melihat pada representasi data kontinyu yaitu datanya bersifat sinambung atau bisa dalam bentuk pecahan maupun desimal. Berikut ini adalah tahapan *Multinomial Naïve Bayes* yang akan ditunjukkan pada Gambar 4.20.

4.1.4.6 *Multinomial Naïve Bayes Training*

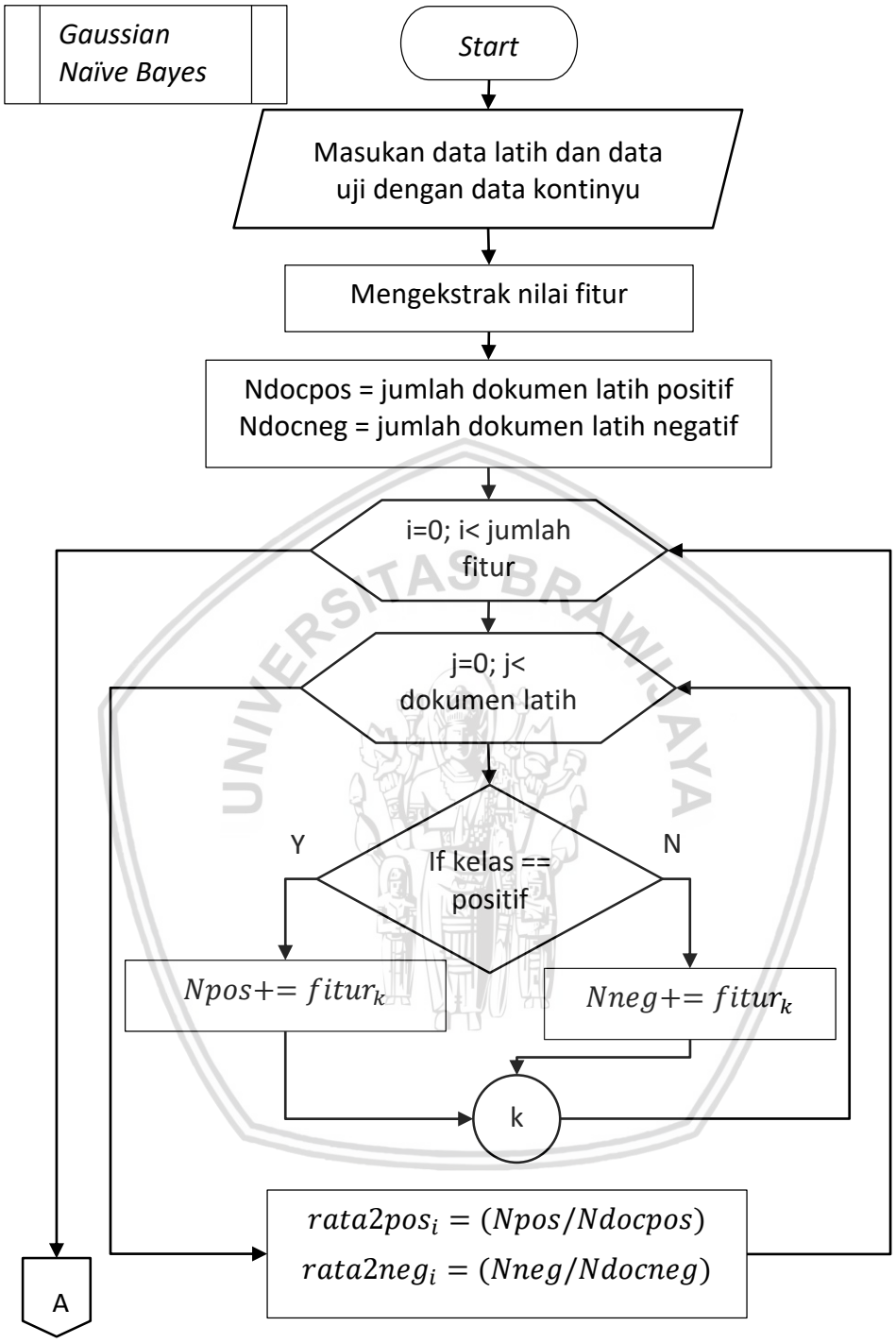
Dalam tahapan distribusi *Multinomial Naïve Bayes Training*, proses perhitungan melihat pada representasi data diskrit yaitu dengan melihat jumlah kemunculan fitur pada dokumen. Berikut ini adalah tahapan *Multinomial Naïve Bayes* pada data latih yang akan ditunjukkan pada Gambar 4.21.

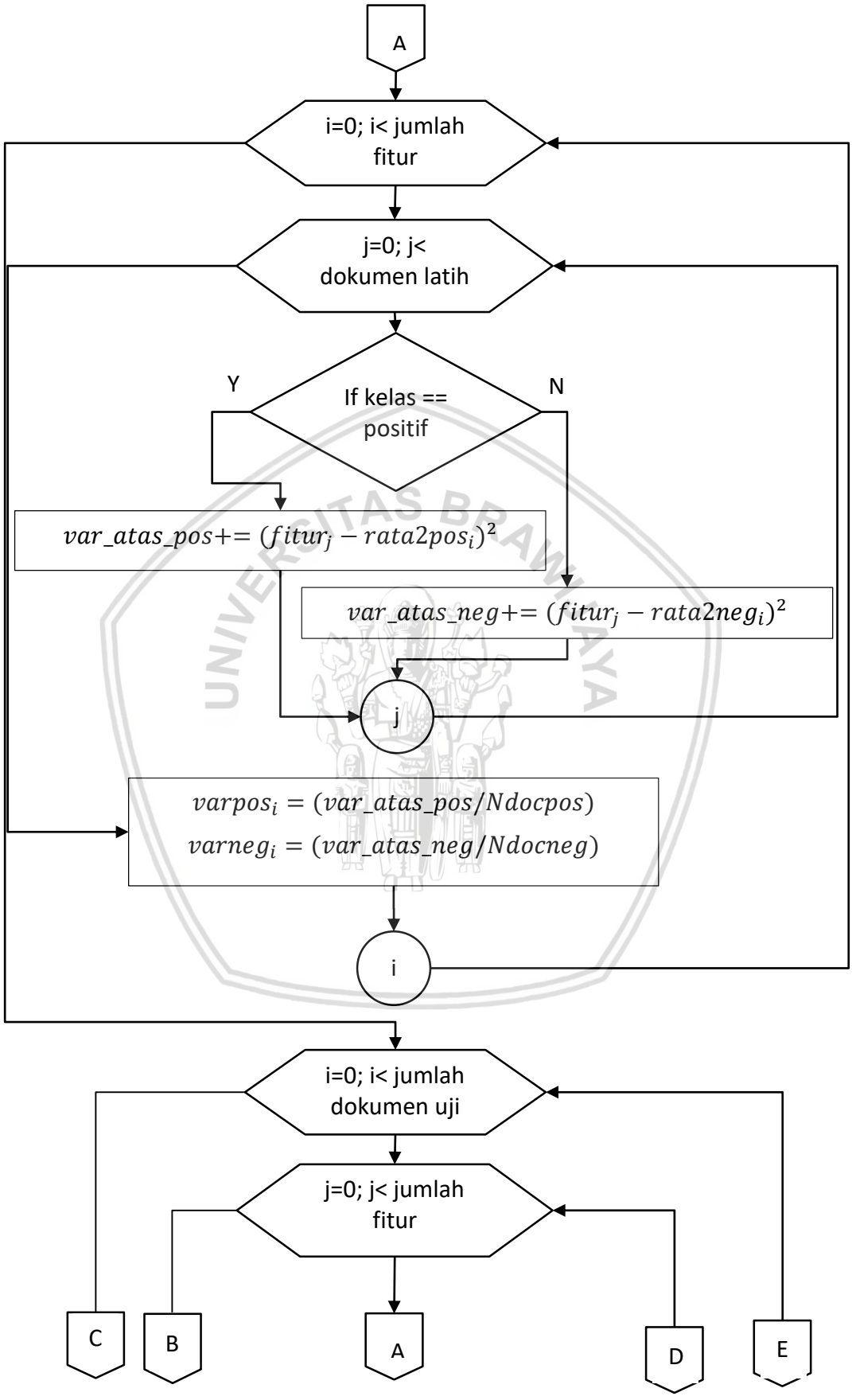


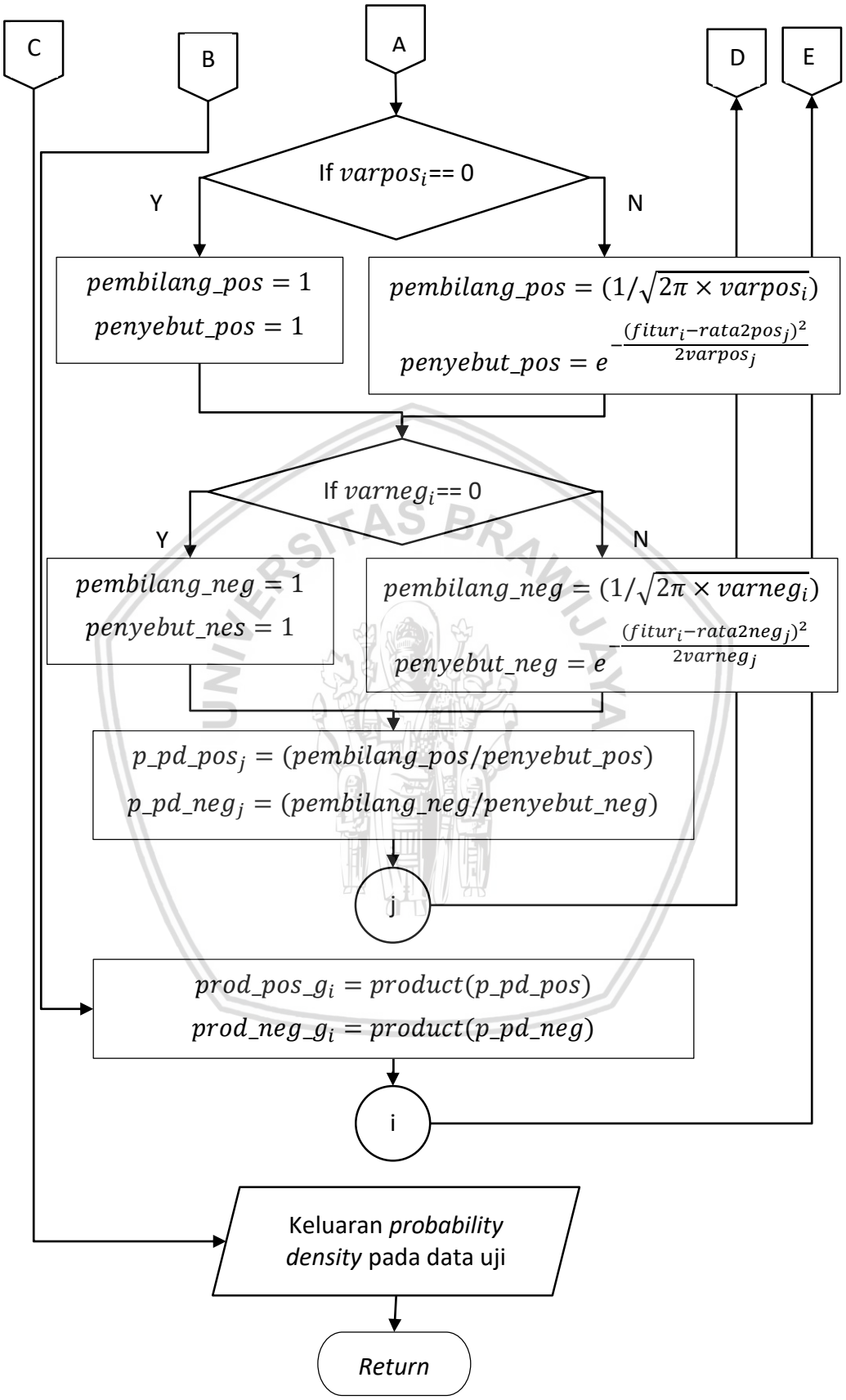
Gambar 4.18 Tahapan pada *Bernoulli Naïve Bayes* untuk Data Latih



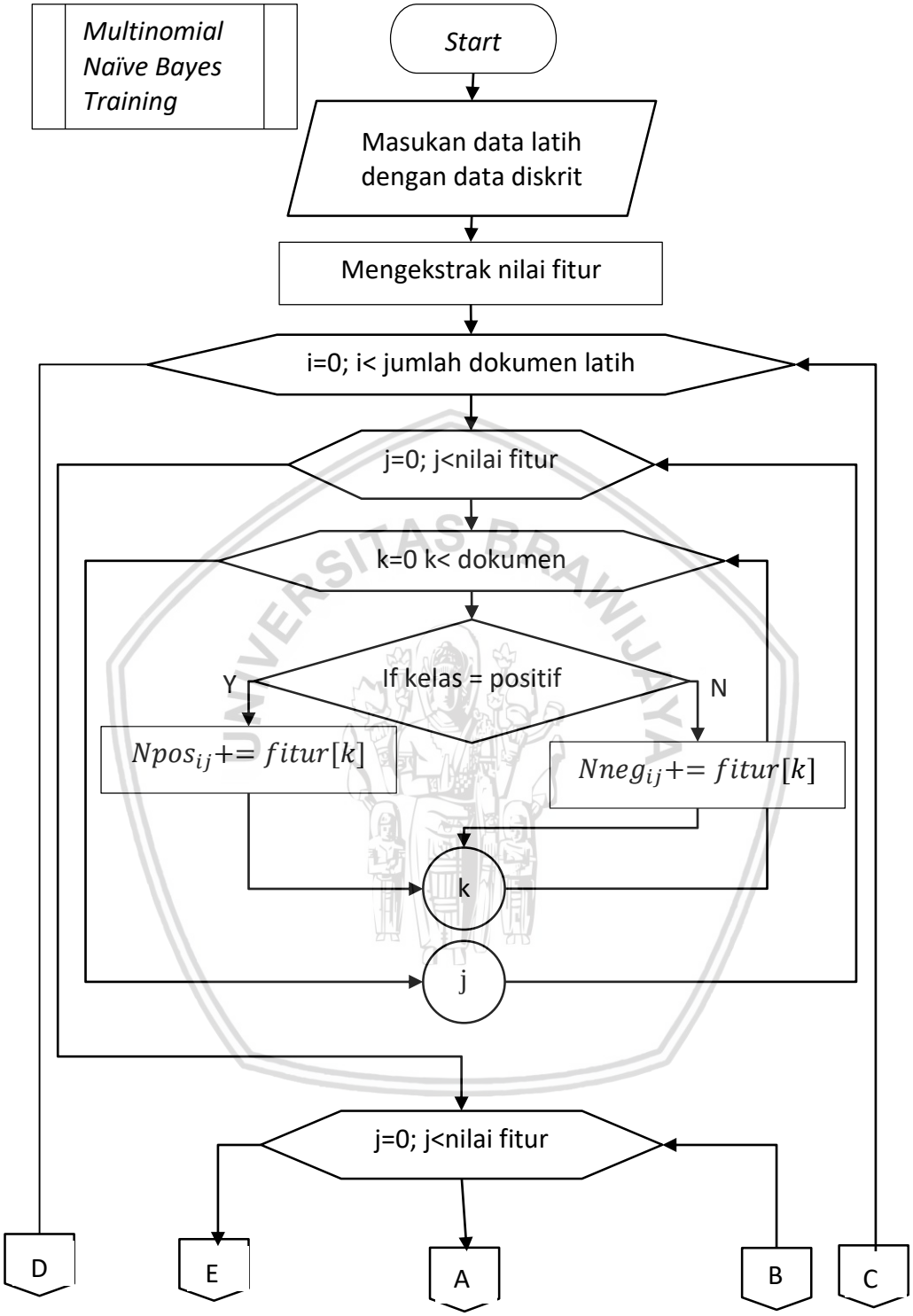
Gambar 4.19 Tahapan pada *Bernoulli Naïve Bayes* untuk Data Uji

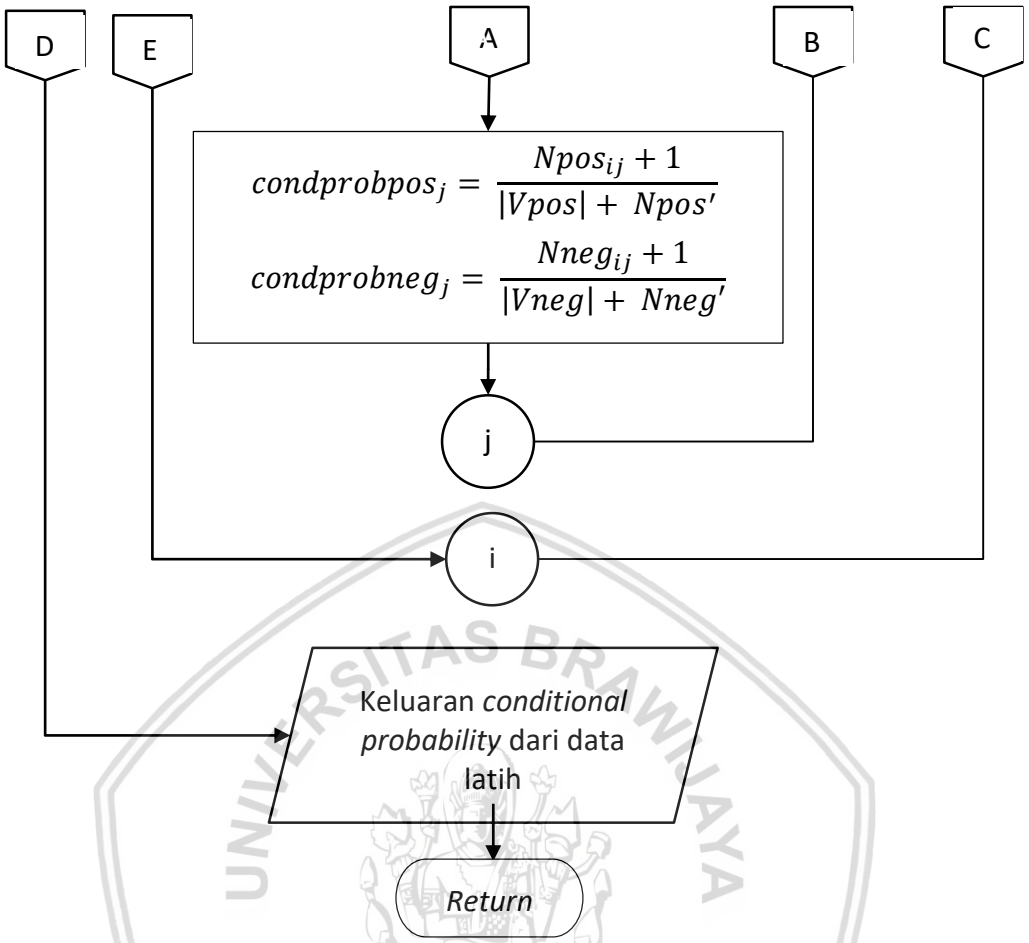

Gaussian
Naïve Bayes
Start
Masukan data latih dan data uji dengan data kontinyu
Mengekstrak nilai fitur
Ndocpos = jumlah dokumen latih positif
Ndocneg = jumlah dokumen latih negatif
i=0; i< jumlah fitur
j=0; j< dokumen latih
Y
If kelas == positif
N
$Npos+= fitur_k$
$Nneg+= fitur_k$
k
$rata2pos_i = (Npos/Ndocpos)$
$rata2neg_i = (Nneg/Ndocneg)$
A


A
i=0; i< jumlah fitur
j=0; j< dokumen latih
Y
If kelas == positif
N
$var\_atas\_pos += (fitur_j - rata2pos_i)^2$
$var\_atas\_neg += (fitur_j - rata2neg_i)^2$
j
$varpos_i = (var\_atas\_pos / Ndocpos)$
$varneg_i = (var\_atas\_neg / Ndocneg)$
i
i=0; i< jumlah dokumen uji
j=0; j< jumlah fitur
C
B
A
D
E



Gambar 4.20 Tahapan pada *Gaussian Naïve Bayes*

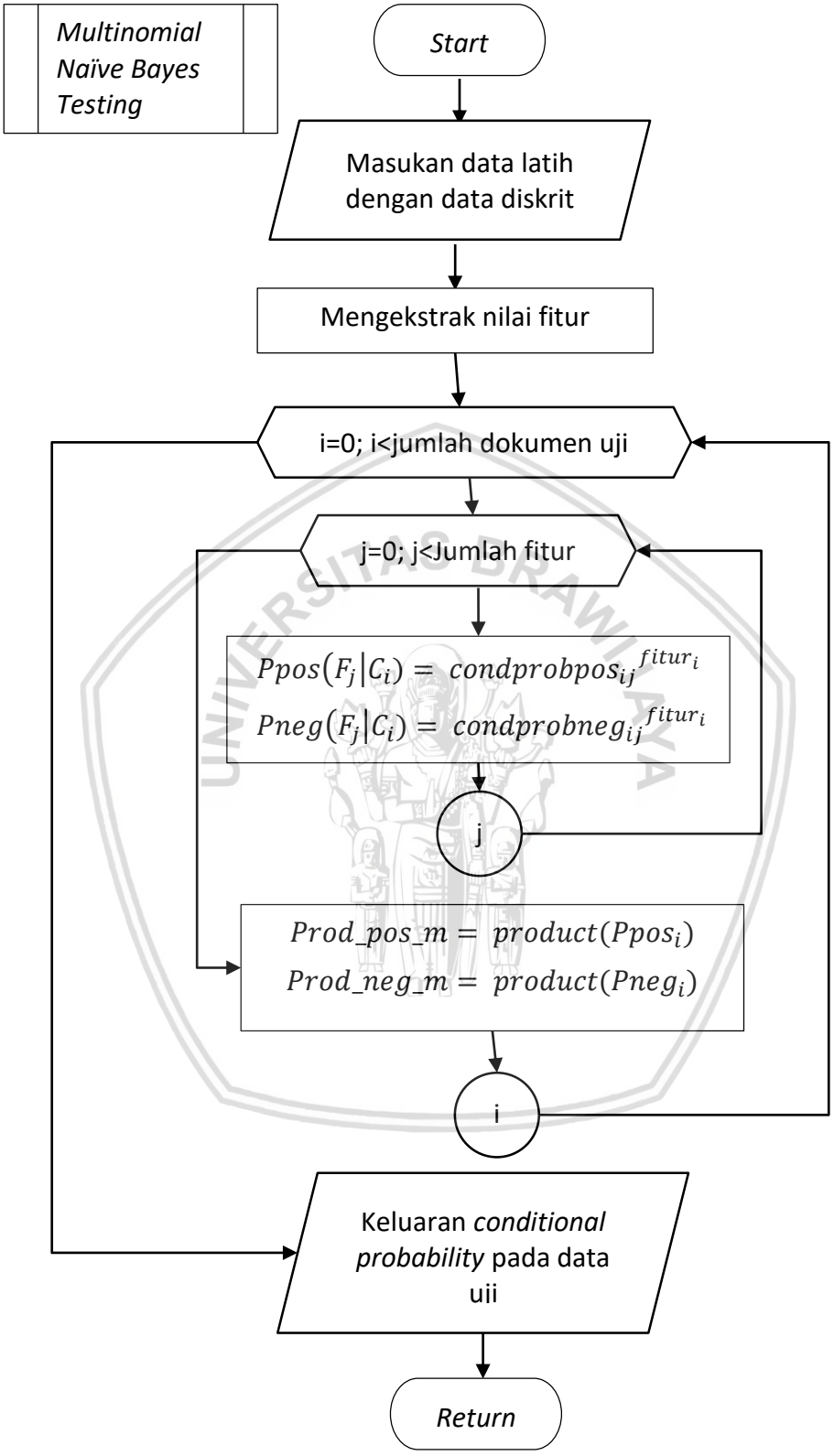

Multinomial Naïve Bayes Training
Start
Masukan data latih dengan data diskrit
Mengekstrak nilai fitur
i=0; i< jumlah dokumen latih
j=0; j<nilai fitur
k=0 k< dokumen
If kelas = positif
Y
N
$Npos_{ij}$+= $fitur[k]$
$Nneg_{ij}$+= $fitur[k]$
k
j
j=0; j<nilai fitur
D
E
A
B
C



Gambar 4.21 Tahapan pada *Multinomial Naïve Bayes* untuk Data Latih

4.1.4.7 *Multinomial Naïve Bayes Testing*

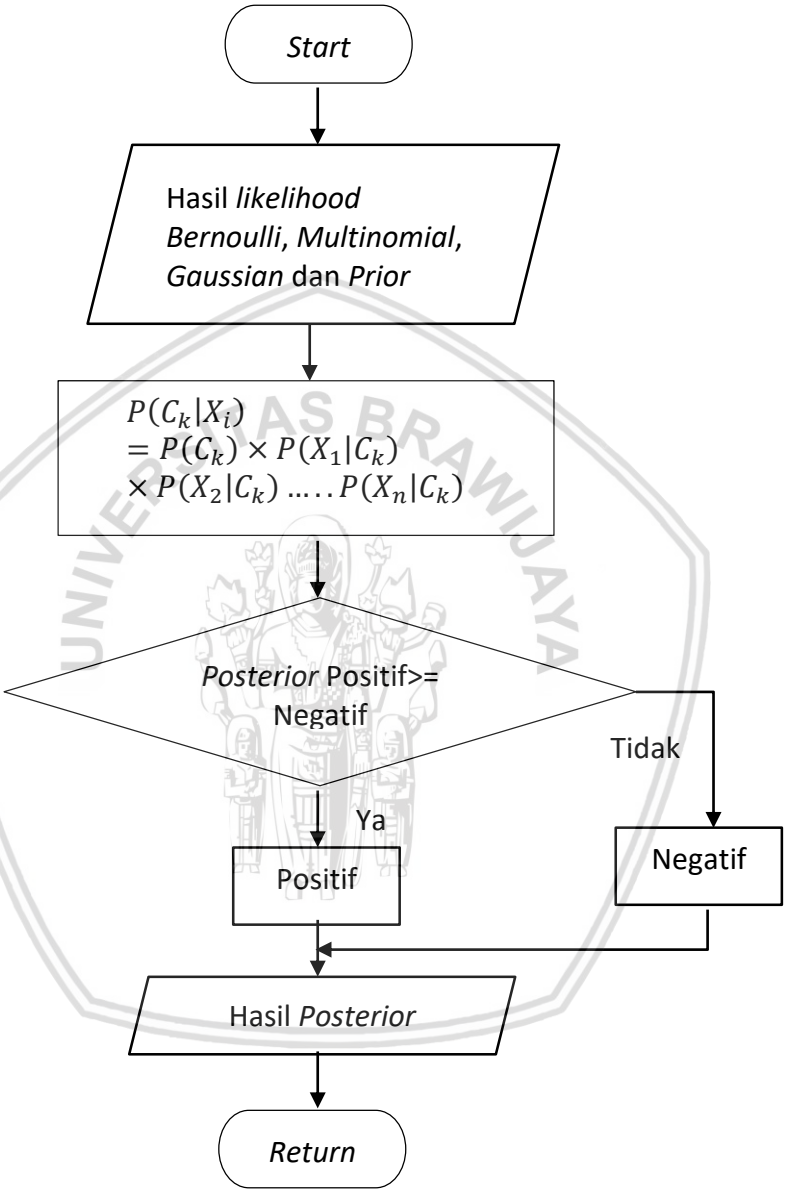
Dalam tahapan distribusi *Multinomial Naïve Bayes Testing*, proses perhitungan melihat pada representasi data diskrit yaitu dengan melihat jumlah kemunculan fitur pada dokumen. Berikut ini adalah tahapan *Multinomial Naïve Bayes* pada data uji yang akan ditunjukkan pada Gambar 4.22.



Gambar 4.22 Tahapan pada *Multinomial Naïve Bayes* untuk Data Uji

4.1.4.8 *Posterior*

Pada tahapan ini adalah tahapan terakhir dalam pengklasifikasian Naïve Bayes untuk mencari hasil akhir klasifikasi dengan cara mengkalikan semua nilai *likelihood* Bernoulli, Multinomial, Gaussian dan nilai *prior*. Berikut ini adalah tahapan *posterior* ditunjukkan pada Gambar 4.23.



Gambar 4.23 Tahapan pada *Posterior*

4.2 Manualisasi

Pada tahap ini akan dilakukan perhitungan manual untuk memudahkan dalam merancang alur program terhadap data latih dan data uji. Beberapa tahapan yang akan dilakukan yaitu manualisasi *ensemble features,* manualisasi *text preprocessing*, manualisasi *Multinomial Naïve Bayes*, manualisasi *Bernoulli Naïve Bayes*, dan *Gaussian Naïve Bayes*. Pada perhitungan ini akan digunakan 4

dokumen sebagai data latih dan 1 dokumen sebagai data uji. Berikut ini adalah sampel data latih dan data uji yang akan ditunjukkan pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Data Latih

No	*Tweet*	Kelas
1.	Dan sekarang dipaksa nonton Warkop DKI Reborn part 2. :"(	Negatif
2.	Saya suka video @YouTube http://youtu.be/XBZq54HJNnI?a Warkop DKI Reborn Full Movie Part 2 Jangkrik Bosssss	Positif
3.	Film Posesif lebih serem dari Pengabdi Setan.	Negatif
4.	BAGUS SEKALI @mira1309 ANDA MEMBUAT SAYA BENAR BENAR HARUS NONTON EIFFEL I'M IN LOVE 2	Positif

Tabel 4.2 Data Uji

No	*Tweet*	Kelas
1.	Kemarin nonton film #Posesif dan ternyata beneran bagus banget	?

4.2.1 Manualisasi *Ensemble Features*

Pada tahap ini akan dilakukan perhitungan *ensemble features* yang meliputi *twitter specific features, textual features*, *part of speech* (*PoS*) *features*, *lexicon based features*, dan *bag of words features*.

4.2.1.1 Manualisasi *Twitter Specific Features*

Pada tahap ini, proses perhitungan F1-F4 akan dinyatakan dalam nilai biner yaitu 1 dan 0. Nilai 1 menunjukkan bahwa *tweet* tersebut mengandung fitur tersebut dan Nilai 0 menunjukkan bahwa *tweet* tersebut tidak mengandung fitur tersebut. Pada Fitur 1 (F1) mendefinisikan apakah *tweet* mengandung *#hashtag* atau tidak, Fitur 2 (F2) mendefinisikan apakah *tweet* mengandung *retweet* (RT) atau tidak, Fitur 3 (F3) mendefinisikan apakah *tweet* mengandung *username* (@) atau tidak, dan Fitur 4 (F4) mendefinisikan apakah *tweet* mengandung tautan *URL* atau tidak. Berikut ini akan ditunjukkan perhitungan *twitter specific features* yang akan ditunjukkan pada Tabel 4.3.

Tabel 4.3 Perhitungan *Twitter Specific Features*

ID \ D	D1	D2	D3	D4	D5
F1	0	0	0	0	1
F2	0	0	0	0	0
F3	0	1	0	1	0
F4	0	1	0	0	0

4.2.1.2 Manualisasi *Textual Features*

Pada tahap ini akan dilakukan proses perhitungan nilai fitur F5-F12 dengan melihat informasi eksplisit pada sebuah *tweet*. Pada Fitur 5 (F5) mendefinisikan jumlah kata pada sebuah *tweet*, Fitur 6 (F6) mendefinisikan rata-rata panjang karakter sebuah kata, Fitur 7 (F7) mendefinisikan jumlah tanda tanya yang terdapat pada sebuah *tweet*, Fitur 8 (F8) mendefinisikan jumlah tanda seru yang terdapat pada sebuah *tweet*, Fitur 9 (F9) mendefinisikan jumlah tanda kutip yang terdapat pada sebuah *tweet*, Fitur 10 (F10) mendefinisikan jumlah kata yang diawali dengan huruf besar pada *tweet*, Fitur 11 (F11) mendefinisikan apakah *tweet* mengandung positif *emoticon* atau tidak, dan Fitur 12 (F12) mendefinisikan apakah *tweet* mengandung negatif *emoticon* atau tidak. Berikut ini adalah perhitungan *textual features* yang akan ditunjukkan pada Tabel 4.4.

Tabel 4.4 Perhitungan *Textual Features*

ID \ D	D1	D2	D3	D4	D5
F5	8	11	7	13	8
F6	6	4,5	5	4,692308	5,5
F7	0	0	0	0	0
F8	0	0	0	0	0
F9	0	0	0	0	0
F10	4	9	4	13	1
F11	0	0	0	0	0
F12	1	0	0	0	0

4.2.1.3 Manualisasi *Part of Speech Features*

Pada tahap ini akan dilakukan proses perhitungan nilai fitur F13-F22 dengan memanfaatkan *Kateglo API* untuk mendapatkan informasi sebuah kata pada *tweet*. Pada Fitur 13 (F13) mendefinisikan jumlah kata benda pada sebuah *tweet*, Fitur 14 (F14) mendefinisikan jumlah kata sifat pada *tweet*, Fitur 15 (F15) mendefinisikan jumlah kata kerja pada *tweet*, Fitur 16 (F16) mendefinisikan jumlah kata keterangan pada *tweet*, dan Fitur 17 (F17) mendefinisikan jumlah *interjection* pada *tweet*. Selanjutnya Fitur 18 (F18) mendefinisikan persentase kata benda pada *tweet*, Fitur 19 (F19) mendefinisikan persentase kata sifat pada *tweet*, Fitur 20 (F20) mendefinisikan persentase kata kerja pada *tweet*, Fitur 21 (F21) mendefinisikan persentase kata keterangan pada *tweet*, dan Fitur 22 (F22) mendefinisikan persentase *interjection* pada *tweet*. Berikut ini adalah perhitungan *part of speech features* yang akan ditunjukkan pada Tabel 4.5.

Tabel 4.5 Perhitungan *Part of Speech Features*

D / ID	D1	D2	D3	D4	D5
F13	1	1	3	2	2
F14	0	1	3	3	1
F15	2	0	0	2	2
F16	0	0	0	0	0
F17	1	0	0	0	1
F18	0,125	0,09090909	0,428571	0,153846	0,25
F19	0	0,09090909	0,428571	0,230769	0,125
F20	0,25	0	0	0,153846	0,25
F21	0	0	0	0	0
F22	0,125	0	0	0	0,125

4.2.1.4 Manualisasi *Lexicon Based Features*

Pada tahap ini akan dilakukan perhitungan nilai fitur F23-F37 dengan menggunakan kamus kata yaitu *sentiwords* dan *boosterwords*. Pada Fitur 23 (F23) mendefinisikan jumlah kata positif pada *tweet*, Fitur 24 (F24) mendefinisikan jumlah kata negatif pada *tweet*, Fitur 25 (F25) mendefinisikan jumlah kata positif dengan kata sifat, Fitur 26 (F26) mendefinisikan jumlah kata negatif dengan kata sifat, Fitur 27 (F27) mendefinisikan jumlah kata positif dengan kata kerja, Fitur 28 (F28) mendefinisikan jumlah kata negatif dengan kata kerja, Fitur 29 (F29) mendefinisikan jumlah kata positif dengan kata keterangan,

dan Fitur 30 (F30) mendefinisikan jumlah kata negatif dengan kata keterangan. Selanjutnya Fitur 31 (F31) mendefinisikan persentase kata positif dengan kata sifat, Fitur 32 (F32) mendefinisikan persentase kata negatif dengan kata sifat, Fitur 33 (F33) mendefinisikan persentase kata positif dengan kata kerja, Fitur 34 (F34) mendefinisikan persentase kata negatif dengan kata kerja, Fitur 35 (F35) mendefinisikan persentase kata positif dengan kata keterangan, Fitur 36 (F36) mendefinisikan persentase kata negatif dengan kata keterangan, dan Fitur 37 (F37) mendefinisikan jumlah kata *intensifier* pada *tweet*. Berikut ini adalah perhitungan *lexicon based features* yang akan ditunjukkan pada Tabel 4.6.

Tabel 4.6 Perhitungan *Lexicon Based Features*

D / ID	D1	D2	D3	D4	D5
F23	0	1	0	3	1
F24	0	0	1	0	0
F25	0	1	1	3	1
F26	0	0	2	0	0
F27	0	0	0	0	0
F28	0	0	0	0	0
F29	0	0	0	0	1
F30	0	0	0	0	0
F31	0	1	0	1	1
F32	0	0	2	0	0
F33	0	0	0	0	0
F34	0	0	0	0	0
F35	0	0	0	0	1
F36	0	0	0	0	0
F37	0	0	0	1	1

4.2.1.5 Manualisasi *Bag of Words Features*

Pada tahap ini, akan dilakukan pra-pemrosesan pada *bag of word features*. Pra-pemrosesan yang akan dilakukan terdiri dari *case folding*, data *cleaning*, *case folding*, *filtering* atau *stopword removal*, dan tokenisasi. Setelah pra-pemrosesan dilakukan, proses perhitungan bobot kata dengan menggunakan *raw term frequency* yaitu menganggap semua *term* sama pentingnya. Berikut ini adalah manualisasi *case folding* yang akan ditunjukkan pada Tabel 4.7.

Setelah proses *case folding* dilakukan, maka selanjutnya akan masuk proses *data cleaning*. Berikut ini adalah manualisasi *data cleaning* yang akan ditunjukkan pada Tabel 4.8. Setelah proses *data cleaning* dilakukan, maka selanjutnya akan masuk proses *stemming* yang akan ditunjukkan pada Tabel 4.9.

Tabel 4.7 Hasil *Case Folding*

Case Folding
Teks
Kemarin nonton film #Posesif dan ternyata beneran bagus banget
Hasil *Case Folding*
Kemarin nonton film #posesif dan ternyata beneran bagus banget

Tabel 4.8 Hasil *Data Cleaning*

Data Cleaning
Hasil *Case Folding*
kemarin nonton film #posesif dan ternyata beneran bagus banget
Hasil *Data Cleaning*
kemarin nonton film posesif dan ternyata beneran bagus banget

Tabel 4.9 Hasil *Stemming*

Stemming
Hasil *Data Cleaning*
kemarin nonton film posesif dan ternyata beneran bagus banget
Hasil *Stemming*
kemarin nonton film posesif dan nyata beneran bagus banget

Setelah proses *stemming* dilakukan, maka selanjutnya akan masuk proses *filtering* atau *stopword removal*. Berikut ini adalah manualisasi *filtering* atau *stopword removal* yang akan ditunjukkan pada Tabel 4.10. Setelah proses *filtering* atau *stopword removal* dilakukan, maka selanjutnya akan masuk proses tokenisasi yang ditunjukkan pada Tabel 4.11.

Tabel 4.10 Hasil *Filtering* atau *Stopword Removal*

Filtering* atau *Stopword Removal
Hasil *Stemming*
kemarin nonton film posesif dan nyata beneran bagus banget
Hasil *Filtering* atau *Stopword Removal*
Kemarin nonton film posesif nyata beneran bagus banget

Tabel 4.11 Hasil Tokenisasi

Tokenisasi
Hasil *Filtering* atau *Stopword Removal*
Kemarin nonton film posesif nyata beneran bagus banget
Hasil Tokenisasi
“Kemarin”, “nonton”, “film”, “posesif” , “nyata”, “beneran”, “bagus”, “banget”

4.2.1.6 Manualisasi *Raw Term Frequency*

Langkah selanjutnya yaitu dengan melakukan perhitungan *raw term frequency*. Perhitungan *raw term frequency* melihat pada semua kemunculan *term* dianggap sama pentingnya. Jika kata ‘sukses’ muncul sebanyak 2 kali maka *raw term frequency* juga sebanyak 2 kali. Berikut ini adalah perhitungan *raw term frequency* yang akan ditunjukkan pada Tabel 4.12.

Tabel 4.12 Perhitungan *Raw Term Frequency*

Term	**D1**	**D2**	**D3**	**D4**	**D5**
sekarang	1	0	0	0	0
paksa	1	0	0	0	0
nonton	1	0	0	1	1
warkop	1	1	0	0	0
dki	1	1	0	0	0
reborn	1	1	0	0	0
part	1	1	0	0	0
saya	0	1	0	1	0
suka	0	1	0	0	0
video	0	1	0	0	0

Term	D1	D2	D3	D4	D5
full	0	1	0	0	0
movie	0	1	0	0	0
bosssss	0	1	0	0	0
Film	0	0	1	0	1
posesif	0	0	1	0	0
lebih	0	0	1	0	0
serem	0	0	1	0	0
pengabdi	0	0	1	0	0
setan	0	0	1	0	0
bagus	0	0	0	1	1
sekali	0	0	0	1	0
anda	0	0	0	1	0
buat	0	0	0	1	0
benar	0	0	0	2	0
harus	0	0	0	1	0
eiffel	0	0	0	1	0
Im	0	0	0	1	0
In	0	0	0	1	0
Love	0	0	0	1	0
kemarin	0	0	0	0	1
ternyata	0	0	0	0	1
banget	0	0	0	0	1

4.2.2 Manualisasi Klasifikasi *Naïve Bayes*

Pada tahapan ini akan dilakukan perhitungan menggunakan klasifikasi naïve bayes. Tahapan perhitungan klasifikasi naïve bayes akan menggunakan dengan tiga jenis distribusi naïve bayes yang terdiri dari *Bernoulli Naïve Bayes*, *Multinomial Naïve Bayes*, dan *Gaussian Naïve Bayes*. Hasil dari manualisasi *ensemble features* akan dikelompokkan berdasarkan bentuk datanya, apakah termasuk data biner, data diskrit, atau data kontinyu. Langkah awal adalah probabilitas *prior* kelas negatif dan kelas positif yaitu 0.5 dan 0.5.

4.2.2.1 Manualisasi *Bernoulli Naïve Bayes*

Pada tahapan ini akan dilakukan perhitungan *Bernoulli Naïve Bayes* dengan melihat pada representasi data biner, apakah termasuk 1 (terdapat fitur) atau 0 (tidak terdapat fitur) pada dokumen. Berikut ini adalah daftar *ensemble features* dengan data biner yang akan ditunjukkan pada Tabel 4.13.

Tabel 4.13 Daftar *Ensemble Features* dengan *Bernoulli Naïve Bayes*

Bernoulli Naive Bayes							
KELAS	**NEGATIF**	**POSITIF**	**NEGATIF**	**POSITIF**	**?**	**Total Negatif Data Latih**	**Total Positif Data Latih**
ID	**D1**	**D2**	**D3**	**D4**	**D5**		
F1	0	0	0	0	1	0	0
F2	0	0	0	0	0	0	0
F3	0	1	0	1	0	0	2
F4	0	1	0	0	0	0	1
F11	0	0	0	0	0	0	0
F12	1	0	0	0	0	1	0

Tabel 4.13 menunjukkan daftar *ensemble features* dengan *Bernoulli Naïve Bayes* yang mana Fitur 1, Fitur 2, Fitur 3, Fitur 4, Fitur 11, dan Fitur 12 termasuk di dalamnya. Untuk mendapatkan total negatif data latih yaitu dengan menjumlahkan total fitur yang ada pada dokumen kelas negatif di data latih, begitu pula dengan total positif data latih. Langkah selanjutnya yaitu dengan menghitung probabilitas *likelihood*. Berikut ini adalah contoh perhitungan probabilitas data latih dengan *Bernoulli Naïve Bayes* pada Fitur 1.

$$P(F1|NEG) = \frac{(N_{ct} + 1)}{(N_c + 2)} = \frac{(0 + 1)}{(2 + 2)} = 0{,}25$$

Tabel 4.14 Perhitungan Probabilitas Data Latih dengan *Bernoulli Naïve Bayes*

ID	P(F|NEG)	P(F|POS)
F1	0,25	0,25
F2	0,25	0,25
F3	0,25	0,75
F4	0,25	0,5
F11	0,25	0,25
F12	0,5	0,25

Tabel 4.14 menunjukkan perhitungan probabilitas data latih dengan *Bernoulli Naïve Bayes*. Selanjutnya, jika sudah mendapatkan nilai probabilitas fitur maka akan menghitung data uji yaitu dokumen 5 (D5) dengan *Bernoulli Naïve Bayes*. Berikut ini adalah contoh perhitungan probabilitas data uji (D5) dengan *Bernoulli Naïve Bayes* pada F1.

$$P(D5|NEG) = \prod_{i=1}^{n} p_{k_i}^{x_i} (1 - p_{ki})^{(1-x_i)} = 0{,}25^{1} (1 - 0{,}25)^{(1-1)} = 0{,}25$$

Tabel 4.15 Perhitungan Probabilitas Data Uji dengan *Bernoulli Naïve Bayes*

ID	P(D5|NEG)	P(D5|POS)
F1	0,25	0,25
F2	0,75	0,75
F3	0,75	0,25
F4	0,75	0,5
F11	0,75	0,75
F12	0,5	0,25
[redacted]	0,0395508	0,0043945

Tabel 4.15 menunjukkan bahwa probabilitas data uji terhadap kelas negatif sebesar 0,0395508 sedangkan probabilitas data uji terhadap kelas positif sebesar 0,0043945 dengan cara mengkalikan nilai semua fitur dari F1-F4 dan F11-F12.

4.2.2.2 Manualisasi *Multinomial Naïve Bayes*

Pada tahapan ini akan dilakukan perhitungan *Multinomial Naïve Bayes* dengan melihat pada representasi data diskrit yaitu melihat jumlah kemunculan suatu fitur atau kata pada dokumen. Berikut ini adalah daftar *ensemble features* dengan data diskrit yang akan ditunjukkan pada Tabel 4.16.

Tabel 4.16 Daftar *Ensemble Features* dengan *Multinomial Naïve Bayes*

Multinomial Naive Bayes							
Kelas	**Negatif**	**Positif**	**Negatif**	**Positif**	**?**	**Total Negatif Data Latih**	**Total Positif Data Latih**
ID	**D1**	**D2**	**D3**	**D4**	**D5**		
F7	0	0	0	0	0	0	0
F8	0	0	0	0	0	0	0
F9	0	0	0	0	0	0	0

Multinomial Naive Bayes							
Kelas	**Negatif**	**Positif**	**Negatif**	**Positif**	**?**	**Total Negatif Data Latih**	**Total Positif Data Latih**
ID	**D1**	**D2**	**D3**	**D4**	**D5**		
F10	4	9	4	13	1	8	22
F13	1	1	3	2	2	4	3
F14	0	1	3	3	1	3	4
F15	2	0	0	2	2	2	2
F16	0	0	0	1	1	0	1
F17	1	0	0	0	1	1	0
F23	0	1	0	3	1	0	4
F24	1	0	1	0	0	2	0
F25	0	1	1	3	1	1	4
F26	0	0	2	0	0	2	0
F27	0	0	0	0	0	0	0
F28	0	0	0	0	0	0	0
F29	0	0	0	0	1	0	0
F30	0	0	0	0	0	0	0
F37	0	0	0	0	1	0	1
sekarang	1	0	0	0	0	1	0
paksa	1	0	0	0	0	1	1
nonton	1	0	0	1	1	1	1
warkop	1	1	0	0	0	1	1
dki	1	1	0	0	0	1	1
reborn	1	1	0	0	0	1	1
part	1	1	0	0	0	1	1
saya	0	1	0	1	0	0	2
suka	0	1	0	0	0	0	1
video	0	1	0	0	0	0	1
full	0	1	0	0	0	0	1
movie	0	1	0	0	0	0	1

Multinomial Naive Bayes							
Kelas	**Negatif**	**Positif**	**Negatif**	**Positif**	**?**	**Total Negatif Data Latih**	**Total Positif Data Latih**
ID	**D1**	**D2**	**D3**	**D4**	**D5**		
bosssss	0	1	0	0	0	0	1
film	0	0	1	0	1	1	0
posesif	0	0	1	0	0	1	0
lebih	0	0	1	0	0	1	0
serem	0	0	1	0	0	1	0
pengabdi	0	0	1	0	0	1	0
setan	0	0	1	0	0	1	0
bagus	0	0	0	1	1	0	1
sekali	0	0	0	1	0	0	1
anda	0	0	0	1	0	0	1
buat	0	0	0	1	0	0	1
benar	0	0	0	2	0	0	2
harus	0	0	0	1	0	0	1
eiffel	0	0	0	1	0	0	1
im	0	0	0	1	0	0	1
in	0	0	0	1	0	0	1
love	0	0	0	1	0	0	1
kemarin	0	0	0	0	1	0	0
ternyata	0	0	0	0	1	0	0
banget	0	0	0	0	1	0	0

Tabel 4.16 menunjukkan daftar *ensemble features* dengan *Multonomial Naïve Bayes* yang mana Fitur 7, Fitur 8, Fitur 9, Fitur 10, Fitur 13, Fitur 14, Fitur 15, Fitur 16, Fitur 17, Fitur 23, Fitur 24, Fitur 25, Fitur 26, Fitur 27, Fitur 28, Fitur 29, Fitur 30, Fitur 37 dan hasil pemrosesan teks termasuk didalamnya. Untuk mendapatkan total negatif data latih yaitu dengan menjumlahkan total fitur yang ada pada dokumen kelas negatif di data latih, begitu pula dengan total positif data latih. Tahapan selanjutnya yaitu dengan menghitung probabilitas *likelihood*. Berikut ini adalah contoh perhitungan probabilitas data latih dengan *Multinomial Naïve Bayes* pada Fitur 7.

$$P(F7|NEG) = \frac{1 + N_k}{|V| + N'} = \frac{(0+1)}{(50+36)} = 0{,}011628$$

Tabel 4.17 Perhitungan Probabilitas Data Latih dengan *Multinomial Naïve Bayes*

ID	P(F|NEG)	P(F|POS)
F7	0,01162791	0,0087719
F8	0,01162791	0,0087719
F9	0,01162791	0,0087719
F10	0,10465116	0,2017544
F13	0,05813953	0,0350877
F14	0,04651163	0,0438596
F15	0,03488372	0,0263158
F16	0,01162791	0,0175439
F17	0,02325581	0,0087719
F23	0,01162791	0,0438596
F24	0,03488372	0,0087719
F25	0,02325581	0,0438596
F26	0,03488372	0,0087719
F27	0,01162791	0,0087719
F28	0,01162791	0,0087719
F29	0,01162791	0,0087719
F30	0,01162791	0,0087719
F37	0,01162791	0,0175439
sekarang	0,02325581	0,0087719
paksa	0,02325581	0,0087719
nonton	0,02325581	0,0175439
warkop	0,02325581	0,0175439
dki	0,02325581	0,0175439
reborn	0,02325581	0,0175439
part	0,02325581	0,0175439

ID	P(F|NEG)	P(F|POS)
saya	0,01162791	0,0263158
suka	0,01162791	0,0175439
video	0,01162791	0,0175439
full	0,01162791	0,0175439
movie	0,01162791	0,0175439
bosssss	0,01162791	0,0175439
film	0,02325581	0,0087719
posesif	0,02325581	0,0087719
lebih	0,02325581	0,0087719
serem	0,02325581	0,0087719
pengabdi	0,02325581	0,0087719
setan	0,02325581	0,0087719
bagus	0,01162791	0,0175439
sekali	0,01162791	0,0175439
anda	0,01162791	0,0175439
buat	0,01162791	0,0175439
benar	0,01162791	0,0263158
harus	0,01162791	0,0175439
eiffel	0,01162791	0,0175439
im	0,01162791	0,0175439
in	0,01162791	0,0175439
love	0,01162791	0,0175439
kemarin	0,01162791	0,0087719
ternyata	0,01162791	0,0087719
banget	0,01162791	0,0087719

Tabel 4.17 menunjukkan perhitungan probabilitas data latih dengan *Multinomial Naïve Bayes*. Langkah selanjutnya, jika sudah mendapatkan nilai probabilitas fitur maka akan menghitung data uji yaitu dokumen 5 (D5) dengan *Multinomial Naïve Bayes*. Berikut ini adalah contoh perhitungan probabilitas data uji (D5) dengan *Multinomial Naïve Bayes* pada F7.

$$P(D5|NEG) = condprob[t]^{[c]} = 0{,}011628^0 = 1$$

Tabel 4.18 Perhitungan Probabilitas Data Uji dengan *Multinomial Naïve Bayes*

ID	P(D5|NEG)	P(D5|POS)
F7	1	1
F8	1	1
F9	1	1
F10	0,1046512	0,2017544
F13	0,0033802	0,0012311
F14	0,0465116	0,0438596
F15	0,0012169	0,0006925
F16	0,0116279	0,0175439
F17	0,0232558	0,0087719
F23	0,0116279	0,0438596
F24	1	1
F25	0,0232558	0,0438596
F26	1	1
F27	1	1
F28	1	1
F29	1	1
F30	1	1
F37	0,0116279	0,0175439
sekarang	1	1
paksa	1	1
nonton	0,0232558	0,0175439
warkop	1	1
dki	1	1
reborn	1	1
part	1	1
saya	1	1
suka	1	1
video	1	1

ID	P(D5|NEG)	P(D5|POS)
full	1	1
movie	1	1
bosssss	1	1
film	0,0232558	0,0087719
posesif	1	1
lebih	1	1
serem	1	1
pengabdi	1	1
setan	1	1
bagus	0,0116279	0,0175439
sekali	1	1
anda	1	1
buat	1	1
benar	1	1
harus	1	1
eiffel	1	1
im	1	1
in	1	1
love	1	1
kemarin	0,0116279	0,0087719
ternyata	0,0116279	0,0087719
banget	0,0116279	0,0087719
	1,683E-28	7,141E-29

Tabel 4.18 menunjukkan bahwa probabilitas data uji terhadap kelas negatif sebesar 1,683E-28 sedangkan probabilitas data uji terhadap kelas positif sebesar 7,141E-29 dengan cara mengkalikan nilai semua fitur dari F7-F10, F13-F17, F23-F30, F37, dan hasil *text preprocessing*.

4.2.2.3 Manualisasi *Gaussian Naïve Bayes*

Pada tahapan ini akan dilakukan perhitungan *Gaussian Naïve Bayes* dengan melihat pada representasi data kontinyu. Berikut ini adalah daftar *ensemble features* dengan data kontinyu yang akan ditunjukkan pada Tabel 4.19.

Tabel 4.19 Daftar *Ensemble Features* dengan *Gaussian Naïve Bayes*

Gaussian Naive Bayes							
Kelas	**Negatif**	**Positif**	**Negatif**	**Positif**	**?**	**Rata-rata Kelas Negatif**	**Rata-rata Kelas Positif**
ID	**D1**	**D2**	**D3**	**D4**	**D5**		
F5	8	11	7	13	8	7,5	12
F6	6	4,5	5	4,692308	5,5	5,5	4,596154
F18	0,125	0,090909	0,428571	0,153846	0,25	0,276785714	0,122377622
F19	0	0,090909	0,428571	0,230769	0,125	0,214285714	0,160839161
F20	0,25	0	0	0,153846	0,25	0,125	0,076923077
F21	0	0	0	0,076923	0,125	0	0,038461538
F22	0,125	0	0	0	0,125	0,0625	0
F31	0	1	0	1	1	0	1
F32	0	0	2	0	0	1	0
F33	0	0	0	0	0	0	0
F34	0	0	0	0	0	0	0
F35	0	0	0	0	0	0	0
F36	0	0	0	0	0	0	0

Tabel 4.19 menunjukkan daftar *ensemble features* dengan *Gaussian Naïve Bayes* yang mana Fitur 5, Fitur 6, Fitur 9, Fitur 18, Fitur 19, Fitur 20, Fitur 21, Fitur 22, Fitur 31, Fitur 32, Fitur 33, Fitur 34, Fitur 35, dan Fitur 36 termasuk di dalamnya. Cara untuk menghitung rata-rata kelas yaitu jumlah nilai fitur pada kelas dibagi dengan jumlah kelas. Berikut ini adalah contoh pada Fitur 5 untuk menghitung rata-rata kelas negatif dan kelas positif.

$$Rata - rata\ kelas\ negatif\ F5\ (\mu_n) = \frac{8+7}{2} = 7{,}5$$

$Rata - rata\ kelas\ positif\ F5\ (\mu_p) = \frac{11+13}{2} = 12$

Tabel 4.20 *Probability Density Function* pada *Gaussian Naïve Bayes*

ID	Variansi Kelas Negatif	Variansi Kelas Positif	P(F|NEG)	P(F|POS)
F5	0,25	1	0,81764454	196,19957
F6	0,25	0,00924559	0,79808688	4,1622051
F18	0,0230389	0,00099027	2,62900892	12,680793
F19	0,04591837	0,00489021	1,86246681	5,7063351
F20	0,015625	0,00591716	3,19264946	5,1879209
F21	0	0,00147929	1	10,375174
F22	0,00390625	0	6,38473281	1
F31	0	0	1	1
F32	1	0	0,5878277	1
F33	0	0	1	1
F34	0	0	1	1
F35	0	0	1	1
F36	#DIV/0!	0	1	1
			38,285941	3180635,9

Tabel 4.20 menunjukkan *probabily density function* pada distribusi *Gaussian Naïve Bayes.* Untuk mengetahui nilai *probabily density function*, maka selain mencari nilai rata-rata kelas juga mencari nilai variansi kelas negatif dan positif. Berikut ini adalah contoh untuk menghitung nilai variansi kelas negatif $(\sigma^2{}_n)$ dan variansi kelas positif $(\sigma^2{}_p)$ pada F5.

Variansi Kelas Negatif:

$$\sigma^2{}_n = \frac{(x_{D1} - \mu_n)^2 + (x_{D3} - \mu_n)^2}{Jumlah\ data\ kelas\ negatif}$$

$$\sigma^2{}_n = \frac{(8 - 7{,}5)^2 + (7 - 7{,}5)^2}{2}$$

$$\sigma^2{}_n = \frac{(0{,}5)^2 + (-0{,}5)^2}{2}$$

$$\sigma^2{}_n = \frac{0{,}25 + 0{,}25}{2}$$

$$\sigma^2{}_n = 0{,}25$$

Variansi Kelas Positif:

$$\sigma^2{}_p = \frac{(x_{D2} - \mu_p)^2 + (x_{D4} - \mu_p)^2}{Jumlah\ data\ kelas\ positif}$$

$$\sigma^2{}_p = \frac{(11 - 12)^2 + (13 - 12)^2}{2}$$

$$\sigma^2{}_p = \frac{(-1)^2 + (1)^2}{2}$$

$$\sigma^2{}_p = \frac{1 + 1}{2}$$

$$\sigma^2{}_p = 1$$

Setelah menghitung nilai variansi kelas negatif dan variansi kelas positif, langkah selanjutnya adalah menghitung *probability density function* pada *Gaussian Naïve Bayes*. Berikut ini adalah contoh perhitungan *probability density function* kelas negatif pada F5.

$$P(F5|NEG) = \frac{1}{\sqrt{2\pi\sigma^2{}_n}} e^{-\frac{(v-\mu_n)^2}{2\sigma^2{}_n}}$$

$$P(F5|NEG) = \frac{1}{\sqrt{2 \times 3{,}14 \times 0{,}25}} \times 2.17^{-\frac{(8-7{,}5)^2}{2\times 0.25}} = 0{,}7980869 \times 2{,}17^{0.03125}$$

$$P(F5|NEG) = 0{,}81764454$$

Langkah selanjutnya adalah menghitung *probability density function* pada kelas negatif dan kelas negative yaitu dengan mengkalikan semua fitur yang ada di dalam *Gaussian Naïve Bayes* maka akan didapatkan *probability density function* pada kelas negatif yaitu sebesar 38,285941 dan pada kelas positif sebesar 3180635,9. Tahapan terakhir pada klasifikasi Naïve Bayes yaitu dengan menghitung probabilitas *posterior* pada kelas negatif dan probabilitas *posterior* kelas positif. Berikut ini adalah perhitungan probabilitas *posterior* kelas negatif dan kelas positif.

$$\boldsymbol{P(NEG|D5) = P(NEG) \times P(D5|NEG)_{Bernoulli} \times P(D5|NEG)_{Multinomial} \times P(PDF|NEG)}$$

$$P(NEG|D5) = 0{,}5 \times 0{,}395508 \times 1{,}957E - 30 \times 38{,}285941 = 1{,}482E - 30$$

$$\boldsymbol{P(POS|D5) = P(POS) \times P(D5|POS)_{Bernoulli} \times P(D5|POS)_{Multinomial} \times P(PDF|POS)}$$

$$P(POS|D5) = 0{,}5 \times 0.0043945 \times 6{,}264E-31 \times 3180635{,}9 = 4{,}387E-27$$

Hasil perhitungan di atas pada probabilitas *posterior* kelas negatif sebesar 1,482E-30, sedangkan pada probabilitas *posterior* kelas positif 4,387E-27. Berdasarkan hasil perhitungan di atas, dapat disimpulkan bahwa data uji yaitu dokumen 5 termasuk kategori kelas positif karena nilai probabilitas *posterior* lebih besar dari kelas negatif.

4.3 Perancangan Pengujian

Pada tahap ini akan dijelaskan mengenai perancangan pengujian untuk mendapatkan hasil klasifikasi yang terdiri dari opini positif atau negatif dengan menggunakan *ensemble features* dan Naïve Bayes dalam analisis sentimen. Pada bagian ini terdapat 7 skenario perancangan pengujian yang akan dilakukan. Berikut ini adalah perancangan pengujian yang digunakan ditunjukkan pada Tabel 4.21.

Tabel 4.21 Perancangan Pengujian

		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif		
	Negatif		
Accuracy			
Precision			
Recall			
F-measure			

4.3.1 Perancangan Pengujian Pengaruh *Twitter Specific Features* terhadap Akurasi Sistem

Pengujian pertama yaitu pengujian pengaruh *Twitter specific features* yaitu fitur yang melihat spesifikasi pada sebuah *tweet* terhadap akurasi sistem. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *Twitter specific features* terhadap akurasi sistem yang ditunjukkan pada Tabel 4.21.

4.3.2 Perancangan Pengujian Pengaruh *Textual Features* terhadap Akurasi Sistem

Pengujian kedua yaitu pengujian pengaruh *textual features* yaitu fitur yang melihat pada informasi eksplisit yang ada pada sebuah *tweet*. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *textual features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.3.3 Perancangan Pengujian Pengaruh *Part of Speech* (*PoS*) *Features* terhadap Akurasi Sistem

Pengujian ketiga yaitu pengujian pengaruh *part of speech* (*pos*) *features* yaitu fitur yang melihat pada informasi kata yang ada pada sebuah *tweet*. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *part of speech features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.3.4 Perancangan Pengujian Pengaruh *Lexicon Based Features* terhadap Akurasi Sistem

Pengujian keempat yaitu pengujian pengaruh *lexicon based features* yaitu fitur dengan memanfaatkan *lexicon* atau kamus kata. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *lexicon based features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.3.5 Perancangan Pengujian Pengaruh *Bag of Words Features* terhadap Akurasi Sistem

Pengujian kelima yaitu pengujian pengaruh *bag of words features* yaitu fitur dari sisi statistik kata. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *bag of words features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.3.6 Perancangan Pengujian Pengaruh *Ensemble Features* tanpa *Bag of Words features* terhadap Akurasi Sistem

Pengujian keenam yaitu pengujian pengaruh *ensemble features* tanpa *bag of words* yaitu gabungan dari beberapa fitur dari sisi semantik kata. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah

perancangan pengujian pengaruh *ensemble features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.3.7 Perancangan Pengujian Pengaruh *Ensemble Features* dan *Bag of Words Features* terhadap Akurasi Sistem

Pengujian ketujuh yaitu pengujian pengaruh *ensemble features* dan *bag of words features*. Pengujian ini dilakukan untuk melihat apakah fitur tersebut mempunyai pengaruh yang signifikan terhadap hasil analisis sentimen atau tidak. Berikut ini adalah perancangan pengujian pengaruh *ensemble features* dan *bag of words features* terhadap akurasi sistem yang akan ditunjukkan pada Tabel 4.21.

4.4 Implementasi *Ensemble Features*

Pada tahapan ini akan dilakukan implementasi kode program ekstraksi fitur dengan menggunakan *ensemble features* dengan menggunakan sebanyak 37 fitur. Pada tahapan ini terdapat 4 jenis tipe fitur yaitu *Twitter specific features, textual features, Part of Speech (PoS) features,* dan *lexicon based features*.

4.4.1 Implementasi *Twitter Specific Features*

Pada tahapan ini dilakukan mendapatkan nilai F1-F4 apakah fitur-fitur tersebut bernilai 1 jika ada pada sebuah dokumen atau 0 jika tidak ada pada sebuah dokumen. Berikut ini adalah implementasi kode program F1 hingga F4.

4.4.1.1 Implementasi F1

Pada tahapan ini akan dilakukan untuk memperoleh nilai F1 apakah sebuah *tweet* tersebut terdapat *hashtag* atau tidak. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.1

```
1  def F1(self):
2      fs = []
3      for doc in self.docs:
4          if '#' in doc[1]:
5              fs.append(1)
6          else:
7              fs.append(0)
8      return fs
```

Kode Program 4.1 Implementasi F1

Penjelasan potongan Kode Program 4.1 tentang implementasi proses F1 dijelaskan sebagai berikut:

1. Pada baris 3-7 merupakan proses pengecekan apakah *tweet* tersebut terdapat *hashtag* atau tidak. Jika terdapat *hashtag* maka akan ditambahkan 1 namun jika tidak terdapat *hashtag* maka akan ditambahkan 0.

4.4.1.2 Implementasi F2

Pada tahapan ini akan dilakukan untuk memperoleh nilai F2 apakah sebuah *tweet* tersebut terdapat *retweet* atau tidak. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.2.

```
1  def F2(self):
2      fs = []
3      for doc in self.docs:
4          if 'RT' in doc[1]:
5              fs.append(1)
6          else:
7              fs.append(0)
8      return fs
```

Kode Program 4.2 Implementasi F2

Penjelasan potongan Kode Program 4.2 tentang implementasi proses F2 dijelaskan sebagai berikut:

1. Pada baris 3-7 merupakan proses pengecekan apakah *tweet* tersebut terdapat *retweet* atau tidak. Jika terdapat *retweet* maka akan ditambahkan 1 namun jika tidak terdapat *retweet* maka akan ditambahkan 0.

4.4.1.3 Implementasi F3

Pada tahapan ini akan dilakukan untuk memperoleh nilai F3 apakah sebuah *tweet* tersebut terdapat *username* atau tidak. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.3.

```
1  def F3(self):
2      fs = []
3      for doc in self.docs:
4          if '@' in doc[1]:
5              fs.append(1)
6          else:
7              fs.append(0)
8      return fs
```

Kode Program 4.3 Implementasi F3

Penjelasan potongan Kode Program 4.3 tentang implementasi proses F3 dijelaskan sebagai berikut:

1. Pada baris 3-7 merupakan proses pengecekan apakah *tweet* tersebut terdapat *username* atau tidak. Jika terdapat *username* maka akan bernilai 1 namun jika tidak terdapat *username* maka akan bernilai 0.

4.4.1.4 Implementasi F4

Pada tahapan ini akan dilakukan untuk memperoleh nilai F4 apakah sebuah *tweet* tersebut terdapat *URL* atau tidak. Untuk melakukan pengecekan dengan menggunakan *regex*. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.4.

```
1   def F4(self):
2       fs = []
3       regex = "\\(?\\b(http://|www[.])[-A-Za-z0-
4   9+&amp;@#/%?=~_()|!:,.;]*[-A-Za-z0-9+&amp;@#/%=~_()|]"
5       for doc in self.docs:
6           if re.search(regex,doc[1]):
7               fs.append(1)
8           else:
9               fs.append(0)
10      return fs
```

Kode Program 4.4 Implementasi F4

Penjelasan potongan Kode Program 4.4 tentang implementasi proses F4 dijelaskan sebagai berikut:

1. Pada baris 3-9 merupakan proses pengecekan apakah *tweet* tersebut terdapat *URL* atau tidak. Dengan adanya regex maka dapat mengecek *URL* pada sebuah *tweet.* Jika terdapat *URL* maka akan ditambahkan 1 namun jika tidak terdapat *URL* maka akan ditambahkan 0.

4.4.2 Implementasi *Textual Features*

Pada tahapan ini dilakukan mendapatkan nilai F5-F12. Berikut ini adalah implementasi kode program F5 hingga F12 yang akan ditunjukkan pada Kode Program 5.5 hingga 5.12.

4.4.2.1 Implementasi F5

Pada tahapan ini akan dilakukan untuk memperoleh nilai F5 untuk menghitung jumlah kata dalam *tweet*. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.5.

```
1   def F5(self):
2       print "F5"
3       fs = []
4       for doc in self.docs:
5           linkRegex = "\\(?\\b(http://|www[.])[-A-Za-z0-
6   9+&amp;@#/%?=~_()|!:,.;]*[-A-Za-z0-9+&amp;@#/%=~_()|]"
7           nolink = re.sub(linkRegex, '', doc[1])
8           stripped = re.sub(r'[^A-Za-z\s]', '', nolink)
9           wordcount = stripped.split()
10          wordlength=len(wordcount)
11          fs.append(wordlength)
12      print fs
13      return fs
```

Kode Program 4.5 Implementasi F5

Penjelasan potongan Kode Program 4.5 tentang implementasi proses F5 dijelaskan sebagai berikut:

1. Pada baris 4 adalah perulangan sebanyak dokumen.
2. Pada baris 5-6 adalah daftar tanda baca yang akan dihilangkan.
3. Pada baris 7 adalah menghilangkan tanda baca pada setiap dokumen.
4. Pada baris 8 adalah menghilangkan karakter selain huruf.
5. Pada baris 9 adalah memecah dokumen menjadi per kata.
6. Pada baris 10 adalah menghitung ada berapa kata pada dokumen tersebut.
7. Pada baris 11 adalah untuk memasukkan jumlah nilai ke dalam *array*.

4.4.2.2 Implementasi F6

Pada tahapan ini akan dilakukan untuk memperoleh nilai F6 untuk menghitung rata-rata panjang karakter pada sebuah kata. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.6.

```
1   def F6(self):
2       print "F6"
3       fs = []
4       for doc in self.docs:
5           wordcount = doc[1].split()
6           wordlength = (len(wordcount))
7           char = (len(doc[1]))
8           avgword = float(char)/float(wordlength)
9           fs.append(round(avgword, 4))
10      print fs
11      return fs
```

Kode Program 4.6 Implementasi F6

Penjelasan potongan Kode Program 4.6 tentang implementasi proses F6 dijelaskan sebagai berikut:

1. Pada baris 4 adalah perulangan sebanyak dokumen.
2. Pada baris 5 adalah untuk memecah dokumen menjadi kata.
3. Pada baris 6 adalah menghitung kata dalam satu dokumen.
4. Pada baris 7 adalah menghitung karakter huruf dalam satu dokumen.
5. Pada baris 8 adalah menghitung rata-rata jumlah huruf per kata dalam satu dokumen.
6. Pada baris 11 adalah untuk memasukkan jumlah nilai ke dalam *array*.

4.4.2.3 Implementasi F7

Pada tahapan ini akan dilakukan untuk memperoleh nilai F7 untuk menghitung jumlah tanda tanya pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.7.

```
1   def F7(self):
2       print "F7"
3       fs = []
4       for doc in self.docs:
5           fs.append(doc[1].count("?"))
6       print fs
7       return fs
```

Kode Program 4.7 Implementasi F7

Penjelasan potongan Kode Program 4.7 tentang implementasi proses F7 dijelaskan sebagai berikut:

1. Pada baris 3 adalah inisialisasi *array* untuk menyimpan nilai F7.
2. Pada baris 4 adalah perulangan untuk setiap dokumen.
3. Pada baris 5 adalah menghitung jumlah tanda tanya pada setiap dokumen lalu disimpan dalam *array* fs.

4.4.2.4 Implementasi F8

Pada tahapan ini akan dilakukan untuk memperoleh nilai F8 untuk menghitung jumlah tanda seru pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.8.

```
1   def F8(self):
2       print "F8"
3       fs = []
4       for doc in self.docs:
5           fs.append(doc[1].count("!"))
6       print fs
7       return fs
```

Kode Program 4.8 Implementasi F8

Penjelasan potongan Kode Program 4.8 tentang implementasi proses F8 dijelaskan sebagai berikut:

1. Pada baris 3 adalah inisialisasi *array* untuk menyimpan nilai F8.
2. Pada baris 4 adalah perulangan untuk setiap dokumen.
3. Pada baris 5 adalah menghitung jumlah tanda seru pada setiap dokumen lalu disimpan dalam *array* fs.

4.4.2.5 Implementasi F9

Pada tahapan ini akan dilakukan untuk memperoleh nilai F9 untuk menghitung jumlah tanda kutip pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.9.

```
1   def F9(self):
2       print "F9"
3       fs = []
4       regex = '"([^"]*)"'
5       for doc in self.docs:
6           b = len(re.findall(regex, doc[1]))
7           fs.append(b)
8       print fs
9       return fs
```

Kode Program 4.9 Implementasi F9

Penjelasan potongan Kode Program 4.9 tentang implementasi proses F9 dijelaskan sebagai berikut:

1. Pada baris 3 adalah inisialisasi *array* untuk menyimpan nilai F9.
2. Pada baris 4 adalah *regex* untuk pengecekan tanda kutip.
3. Pada baris 5 adalah perulangan sebanyak jumlah dokumen.
4. Pada baris 6-7 adalah menghitung jumlah tanda kutip menggunakan *regex* pada setiap dokumen lalu disimpan pada *array*.

4.4.2.6 Implementasi F10

Pada tahapan ini akan dilakukan untuk memperoleh nilai F10 untuk menghitung jumlah kata yang diawali huruf besar pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.10.

```
1   def F10(self):
2       print "F10"
3       fs = []
4       wordUppercase = "\\b([A-Z][A-Z]+|[A-Z][a-z]+|[A-Z][0-9]+|[A-
5   Z][a-z0-9]+){1,}\\b"
6       for doc in self.docs:
7           b = len(re.findall(wordUppercase, doc[1]))
8           fs.append(b)
9       print fs
10      return fs
```

Kode Program 4.10 Implementasi F10

Penjelasan potongan Kode Program 4.10 tentang implementasi proses F10 dijelaskan sebagai berikut:

1. Pada baris 3 adalah inisialisasi *array* untuk menyimpan nilai F10.
2. Pada baris 4-5 adalah *regex* untuk pengecekan kata yang diawali huruf besar.
3. Pada baris 6 adalah perulangan untuk setiap dokumen.
4. Pada baris 7-8 adalah menghitung jumlah kata-kata yang diawali dengan huruf besar lalu disimpan ke dalam *array*.

4.4.2.7 Implementasi F11

Pada tahapan ini akan dilakukan untuk memperoleh nilai F11 untuk mengecek apakah ada *emoticon* positif pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.11.

```
1   def F11(self):
2       print "F11"
3       with open("emoticon_id.txt", "rb") as f:
4           emots = csv.reader(f, delimiter='|')
5           emoarray = []
6           for a in emots:
7               emoarray.append(a)
8           fs = []
9           for doc in self.docs:
10              print "doc > ", doc
11              found = False
12              for emot in emoarray:
13                  if emot[1] is not '' and emot[0] in doc[1] and
14  int(emot[1]) > 0:
15                      fs.append(1)
16                      found = True
17                      print "ketemu emopos > ", emot[0]
18                      break
19              if not found:
20                  fs.append(0)
21      print fs
22      return fs
```

Kode Program 4.11 Implementasi F11

Penjelasan potongan Kode Program 4.11 tentang implementasi proses F11 dijelaskan sebagai berikut:

1. Pada baris 3 adalah membuka *file* txt yang berisi daftar *emoticon*.
2. Pada baris 4 adalah membaca isi *file* txt dengan pemisah tanda garis tegak.
3. Pada baris 5 adalah inisialisasi *array* untuk menyimpan *emoticon*.
4. Pada baris 6-7 adalah memasukkan *emoticon-emoticon* dari txt ke dalam *array.*

5. Pada baris 8 adalah inisialisasi *array* untuk menyimpan nilai ada tidaknya *emoticon* pada suatu dokumen.
6. Pada baris 11-20 adalah pencarian *emoticon* dalam dokumen yang mempunyai nilai dan nilai *emoticon*-nya lebih dari 0 maka nilai fiturnya menjadi 1 dan pencarian berhenti dan dilanjutkan ke dokumen selanjutnya. Namun, apabila tidak ditemukan maka nilai fiturnya menjadi 0.

4.4.2.8 Implementasi F12

Pada tahapan ini akan dilakukan untuk memperoleh nilai F12 untuk mengecek apakah ada *emoticon* negatif pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.12.

```
1   def F12(self):
2       print "F12"
3       with open("emoticon_id.txt", "rb") as f:
4           emots = csv.reader(f, delimiter='|')
5           emoarray = []
6           for a in emots:
7               emoarray.append(a)
8           fs = []
9           for doc in self.docs:
10              print "doc > ", doc
11              found = False
12              for emot in emoarray:
13                  if emot[1] is not '' and emot[0] in doc[1] and
14  int(emot[1]) < 0:
15                      fs.append(1)
16                      found = True
17                      break
18              if not found:
19                  fs.append(0)
20      print fs
21      return fs
```

Kode Program 4.12 Implementasi F12

Penjelasan potongan Kode Program 4.12 tentang implementasi proses F12 dijelaskan sebagai berikut:

1. Pada baris 3 adalah membuka *file* txt yang berisi daftar *emoticon*.
2. Pada baris 4 adalah membaca isi *file* txt dengan pemisah tanda garis tegak.
3. Pada baris 5 adalah inisialisasi *array* untuk menyimpan nilai F12.
4. Pada baris 6-7 adalah memasukkan *emoticon-emoticon* dari txt ke dalam *array.*
5. Pada baris 8 adalah inisialisasi *array* untuk menyimpan nilai ada tidaknya *emoticon* pada suatu dokumen.
6. Pada baris 11-19 adalah pencarian *emoticon* dalam dokumen yang mempunyai nilai dan nilai *emoticon*-nya kurang dari 0 maka nilai fiturnya menjadi 1 dan pencarian berhenti. Namun, apabila tidak ditemukan maka nilai fiturnya menjadi 0.

4.4.3 Implementasi *Part of Speech Features*

Pada tahapan ini dilakukan mendapatkan nilai F13-F22. Berikut ini adalah implementasi kode program F13 hingga F22 yang akan ditunjukkan pada Kode Program 4.13 dan 4.14.

4.4.3.1 Implementasi FPOS (F13 hingga F17)

Pada tahapan ini akan dilakukan untuk memperoleh nilai F13, F14, F15, F16, dan F17 untuk menghitung jumlah kata benda, kata sifat, kata kerja, kata keterangan, dan kata seru secara berurutan. Berikut ini adalah implementasi FPOS yang akan ditunjukkan pada Kode Program 4.13.

```
1   def FPOS(self):
2       print "FPOS"
3       fs_noun = []
4       fs_adj = []
5       fs_verb = []
6       fs_adv = []
7       fs_i = []
8       i=0
9       for doc in self.docs:
10          i+=1
11          if i%100 is 0:
12              time.sleep(3)
13          print doc
14          jumlah_n = 0
15          jumlah_a = 0
16          jumlah_v = 0
17          jumlah_adv = 0
18          jumlah_i = 0
19          tokenize = lambda doc: doc.split(" ")
20          kata = tokenize(doc[1].decode('utf-8', 'ignore'))
21          for w in kata:
22              response =
23  requests.get("http://kateglo.com/api.php?format=json&phrase=" + w)
24              try:
25                  json_object = json.loads(response.text)
26                  kelas = json_object["kateglo"]["lex_class"]
27                  if kelas == "n":
28                      jumlah_n += 1
29                  if kelas == "adj":
30                      jumlah_a += 1
31                  if kelas == "v":
32                      jumlah_v += 1
33                  if kelas == "adv":
34                      jumlah_adv += 1
35                  if kelas == "i":
36                      jumlah_i += 1
37              except ValueError as e:
38                  pass
39              time.sleep(0.5)
40          print "jumlah n = ",(jumlah_n)
41          print "jumlah a = ",(jumlah_a)
42          print "jumlah v = ",(jumlah_v)
43          print "jumlah adv = ",(jumlah_adv)
44          print "jumlah i = ",(jumlah_i)
45          fs_noun.append(jumlah_n)
46          fs_adj.append(jumlah_a)
47          fs_verb.append(jumlah_v)
48          fs_adv.append(jumlah_adv)
```

```
49              fs_i.append(jumlah_i)
50          self.fs_noun = fs_noun
51          self.fs_adj = fs_adj
52          self.fs_verb = fs_verb
53          self.fs_adv = fs_adv
54          self.fs_i = fs_i
55          print "fs_noun = ",fs_noun
56          print "fs_adj = ",fs_adj
57          print "fs_verb = ",fs_verb
58          print "fs_adv = ",fs_adv
59          print "fs_i = ",fs_i
60          return fs_noun, fs_adj, fs_verb, fs_adv, fs_i
```

Kode Program 4.13 Implementasi FPOS

Penjelasan potongan Kode Program 5.13 tentang implementasi FPOS dijelaskan sebagai berikut:

1. Pada baris 3-7 adalah inisialisasi *array* untuk menyimpan nilai FPOS.
2. Pada baris 8-12 adalah setiap dokumen dengan kelipatan 100 program akan mengalami jeda selama 3 detik.
3. Pada baris 14-18 adalah inisialisasi jumlah kata benda, kata sifat, kata kerja, kata keterangan, dan kata seru.
4. Pada baris 19 adalah memecah kata pada sebuah dokumen.
5. Pada baris 20 adalah menghilangkan kode utf-8 dari dokumen.
6. Pada baris 21-39 adalah perulangan sebanyak kata pada setiap dokumen. Lalu, pengecekan kata di web kateglo dan menghitung jumlah kata benda, kata sifat, kata kerja, kata keterangan, dan kata seru. Namun, apabila tidak dapat terhubung dengan kateglo maka tidak dilakukan pengecekan. Setelah pengecekan setiap kata program akan melakukan jeda selama 0.5 detik.
7. Pada baris 45-49 adalah memasukkan nilai FPOS pada *array*.

4.4.3.2 Implementasi FPercentagePOS (F18 hingga F22)

Pada tahapan ini akan dilakukan untuk memperoleh nilai F18, F19, F20, F21, dan F22 untuk menghitung persentase jumlah kata benda, kata sifat, kata kerja, kata keterangan, dan kata seru secara berurutan. Berikut ini adalah implementasi FPercentagePOS yang akan ditunjukkan pada Kode Program 4.14.

```
1   def FpercentageofPOS(self,fsPOS):
2       print "FpercentageofPOS"
3       fs = []
4       F5 = self.F5()
5       for i in range(len(fsPOS)):
6           if F5[i] != 0:
7               percentage = (fsPOS[i] / F5[i])
8               fs.append(percentage)
9       print fs
10      return fs
```

Kode Program 4.14 FPercentagePOS

Penjelasan potongan Kode Program 5.14 tentang implementasi FPercentageePOS dijelaskan sebagai berikut:

1. Pada baris 4 adalah inisialisasi dengan mengakses nilai yang ada pada F5.
2. Pada baris 5-7 adalah perulangan sebanyak dokumen dan menghitung persentase dari setiap POS per jumlah kata dalam dokumen.

3. Pada baris 8 adalah memasukkan nilai persentase POS ke dalam *array*.

4.4.4 Implementasi *Lexicon Based Features*

Pada tahapan ini dilakukan mendapatkan nilai F23-F37. Berikut ini adalah implementasi kode program F23 hingga F37 yang akan ditunjukkan pada Kode Program 4.15 hingga 4.17.

4.4.4.1 Implementasi FLexicon (F23 hingga F30)

Pada tahapan ini akan dilakukan untuk memperoleh nilai F23, F24, F25, F26, F27, F28, F29, dan F30 untuk menghitung jumlah kata positif, kata negatif, jumlah kata POS *adjective* yang bersifat positif dan negatif, jumlah kata POS *verb* yang bersifat positif dan negatif, serta jumlah kata POS *adverb* yang bersifat positif dan negatif secara berurutan. Berikut ini adalah implementasi FLexicon yang akan ditunjukkan pada Kode Program 4.15.

```
1   def FLexicon(self):
2       print "FLexicon"
3       with open("sentiwords_id.txt","rb") as f:
4           wordsReader = csv.reader(f, delimiter = ":")
5           words = []
6           for word in wordsReader:
7               words.append(word)
8           fspos = []
9           fsneg = []
10          fs_posadj = []
11          fs_negadj = []
12          fs_posverb = []
13          fs_negverb = []
14          fs_posadv = []
15          fs_negadv = []
16          i = 0
17          for doc in self.docs:
18              print doc
19              i += 1
20              if i % 100 is 0:
21                  time.sleep(3)
22              jumlah_pos = 0
23              jumlah_neg = 0
24              jumlah_posadj = 0
25              jumlah_negadj = 0
26              jumlah_posverb = 0
27              jumlah_negverb = 0
28              jumlah_posadv = 0
29              jumlah_negadv = 0
30              tokenize = lambda doc: doc.lower().split(" ")
31              kata = tokenize(doc[1].decode('utf-8', 'ignore'))
32              for k in kata:
33                  for word in words:
34                      if k == word[0] and word[1] is not '' and
35  int(word[1]) > 0:
36                          jumlah_pos += 1
37                      elif k == word[0] and word[1] is not '' and
38  int(word[1]) < 0:
39                          jumlah_neg += 1
40              response =
41  requests.get("http://kateglo.com/api.php?format=json&phrase=" + k)
42              try:
43                  json_object = json.loads(response.text)
```

```
44                      kelas = json_object["kateglo"]["lex_class"]
45                      if kelas == 'adj':
46                          for word in words:
47                              if k == word[0] and word[1] is not ''
48  and int(word[1]) > 0:
49                                  jumlah_posadj += 1
50                              elif k == word[0] and word[1] is not
51  '' and int(word[1]) < 0:
52                                  jumlah_negadj += 1
53                      if kelas == 'v':
54                          for word in words:
55                              if k == word[0] and word[1] is not ''
56  and int(word[1]) > 0:
57                                  jumlah_posverb += 1
58                              elif k == word[0] and word[1] is not
59  '' and int(word[1]) < 0:
60                                  jumlah_negverb += 1
61                      if kelas == 'adv':
62                          for word in words:
63                              if k == word[0] and word[1] is not ''
64  and int(word[1]) > 0:
65                                  jumlah_posadv += 1
66                              elif k == word[0] and word[1] is not
67  '' and int(word[1]) < 0:
68                                  jumlah_negadv += 1
69                  except ValueError as e:
70                      pass
71                  time.sleep(0.5)
72              print "jumlah_pos = ",(jumlah_pos)
73              print "jumlah_neg = ",(jumlah_neg)
74              print "jumlah_posadj = ",(jumlah_posadj)
75              print "jumlah_negadj = ",(jumlah_negadj)
76              print "jumlah_posverb = ",(jumlah_posverb)
77              print "jumlah_negverb = ",(jumlah_negverb)
78              print "jumlah_posadv = ",(jumlah_posadv)
79              print "jumlah_negadv = ",(jumlah_negadv)
80              fspos.append(jumlah_pos)
81              fsneg.append(jumlah_neg)
82              fs_posadj.append(jumlah_posadj)
83              fs_negadj.append(jumlah_negadj)
84              fs_posverb.append(jumlah_posverb)
85              fs_negverb.append(jumlah_negverb)
86              fs_posadv.append(jumlah_posadv)
87              fs_negadv.append(jumlah_negadv)
88      print "fspos = ",fspos
89      print "fsneg = ",fsneg
90      print "fs_posadj = ",fs_posadj
91      print "fs_negadj = ",fs_negadj
92      print "fs_posverb = ",fs_posverb
93      print "fs_negverb = ",fs_negverb
94      print "fs_posadv = ",fs_posadv
95      print "fs_negadv = ",fs_negadv
96      return fspos, fsneg, fs_posadj, fs_negadj, fs_posverb,
97  fs_negverb, fs_posadv, fs_negadv
```

Kode Program 4.15 Implementasi FLexicon

Penjelasan potongan Kode Program 4.15 tentang implementasi FLexicon dijelaskan sebagai berikut:

1. Pada baris 3 adalah membuka *file* txt yang berisi daftar kata-kata yang memiliki sentiment positif dan sentiment negatif.
2. Pada baris 4 adalah membaca isi *file* txt dengan pemisah tanda garis tegak.
3. Pada baris 5-7 adalah perulangan untuk memasukkan *file* txt ke dalam *array*.

4. Pada baris 8-15 adalah inisialisasi *array* untuk menyimpan nilai FLexicon.
5. Pada baris 20-21 adalah setiap dokumen dengan kelipatan 100, program akan mengalami jeda selama 3 detik.
6. Pada baris 22-29 adalah inisialisasi jumlah kata positif, kata negatif, jumlah kata POS *adjective* yang bersifat positif dan negatif, jumlah kata POS *verb* yang bersifat positif dan negatif, serta jumlah kata POS *adverb* yang bersifat positif dan negatif.
7. Pada baris 30 adalah memecah kata pada sebuah dokumen.
8. Pada baris 31 adalah menghilangkan kode utf-8 dari dokumen.
9. Pada baris 32-39 adalah pengecekan kata dari dokumen untuk disamakan dengan kata yang ada di *file* txt. Jika kata tersebut sama, nilai sentimen kata tersebut memiliki nilai, dan nilai tersebut lebih dari 0, maka jumlah kata positif bertambah 1. Namun, jika nilai tersebut kurang dari 0, maka jumlah kata negatif bertambah 1.
10. Pada baris 40-71 adalah perulangan sebanyak kata pada *file* txt. Lalu, pengecekan kata di web kateglo dan menghitung jumlah jumlah kata POS *adjective* yang bersifat positif dan negatif, jumlah kata POS *verb* yang bersifat positif dan negatif, serta jumlah kata POS *adverb* yang bersifat positif dan negatif. Namun, apabila tidak dapat terhubung dengan kateglo maka tidak dilakukan pengecekan. Setelah pengecekan setiap kata program akan melakukan jeda selama 0.5 detik.
11. Pada baris 80-87 adalah memasukkan nilai FLexicon pada *array*.

4.4.4.2 Implementasi FPercentageLexicon (F31 hingga F36)

Pada tahapan ini akan dilakukan untuk memperoleh nilai F31, F32, F33, F34, F35, dan F36 untuk menghitung persentase jumlah kata POS *adjective* yang bersifat positif dan negatif, jumlah kata POS *verb* yang bersifat positif dan negatif, serta jumlah kata POS *adverb* yang bersifat positif dan negatif secara berurutan. Berikut ini adalah implementasi FPercentagePOS yang akan ditunjukkan pada Kode Program 4.16.

```
1   def FpercentageLexicon(self, fs_lexicon, fs_lexicon_pos):
2       print "FpercentageLexicon"
3       fs_p_lexicon = []
4       for i in range(len(fs_lexicon_pos)):
5           percentage = 0
6           if fs_lexicon[i] != 0:
7               percentage = (fs_lexicon_pos[i] / fs_lexicon[i])
8           fs_p_lexicon.append(percentage)
9       print fs_p_lexicon
10      return fs_p_lexicon
```

Kode Program 4.16 Implementasi FPercentageLexicon

Penjelasan potongan Kode Program 4.16 tentang implementasi FPercentageLexicon dijelaskan sebagai berikut:

1. Pada baris 3 adalah inisialisasi *array* untuk menyimpan nilai FPercentageLexicon.

2. Pada baris 4-7 adalah perulangan sebanyak dokumen. Kemudian, menghitung persentase fitur *lexicon* POS terhadap fitur *lexicon*.
3. Pada baris 8 adalah memasukkan nilai persentase lexicon ke dalam *array*.

4.4.4.3 Implementasi F37

Pada tahapan ini akan dilakukan untuk memperoleh nilai F37 untuk menghitung jumlah kata *intensifier* pada sebuah dokumen. Berikut ini adalah kode program yang akan ditunjukkan pada Kode Program 4.17.

```
1   def F37(self):
2       print "F37"
3       with open("boosterwords_id.txt", "rb") as f:
4           wordsReader = csv.reader(f, delimiter =":")
5           words = []
6           for word in wordsReader:
7               words.append(word)
8           fs = []
9           for doc in self.docs:
10              tokenize = lambda doc: doc.lower().split(" ")
11              katakata = tokenize(doc[1].decode('utf-8', 'ignore'))
12              jumlah_intensifier =0
13              for kata in katakata:
14                  for word in words:
15                      if kata == word[0]:
16                          jumlah_intensifier+=1
17                          break
18              fs.append(jumlah_intensifier)
19      print fs
20      return fs
```

Kode Program 4.17 Implementasi F37

Penjelasan potongan Kode Program 4.17 tentang implementasi proses 37 dijelaskan sebagai berikut:

1. Pada baris 3 adalah membuka *file* txt yang berisi daftar kata *intensifier*.
2. Pada baris 4 adalah membaca isi *file* txt dengan pemisah tanda titik dua.
3. Pada baris 5 adalah inisialisasi *array* untuk menyimpan kata *intensifier*.
4. Pada baris 6-7 adalah memasukkan kata *intensifier* dari txt ke dalam *array.*
5. Pada baris 8-9 adalah inisialisasi *array* untuk menyimpan nilai kata *intensifier* dan melakukan perulangan sebanyak jumlah dokumen.
6. Pada baris 10 adalah memecah kata pada sebuah dokumen.
7. Pada baris 11 adalah menghilangkan kode utf-8 dari dokumen.
8. Pada baris 12-17 adalah pengecekan kata dari dokumen untuk disamakan dengan kata yang ada di *file* txt. Jika kata tersebut sama, maka jumlah kata *intensifier* bertambah 1. Jika kata tersebut tidak sama, maka kata *intensifier* bernilai 0.

4.5 Implementasi *Bag of Words Features*

Pada tahapan ini akan dilakukan implementasi kode program ekstraksi fitur dengan menggunakan *bag of words features* melalui beberapa tahapan, yaitu *case folding, data cleaning, stemming,* tokenisasi, dan *raw term frequency.*

4.5.1 Implementasi *Case Folding*

Pada tahapan *case folding* ini bertujuan untuk mengubah semua karakter huruf menjadi semua huruf kecil. Berikut ini adalah implementasi *case folding* yang akan ditunjukkan pada Kode Program 4.18.

```
1   def Bag_of_Words(self, listkata):
2       print "Fitur Bag of Words"
3       print "# casefolding"
4       for doc in listkata:
5           casefolding = doc[1].lower()
6           linkRegex =  r'\w+:\/{2}[\d\w-]+(\.[\d\w-
7   ]+)*(?:(?:\/[^\s/]*))*'
8
9           nolink = re.sub(linkRegex, '', casefolding)
10          doc[1] = nolink
            print doc[1]
```

Kode Program 4.18 Implementasi *Case Folding*

Penjelasan potongan Kode Program 4.18 tentang implementasi proses *case folding* dijelaskan sebagai berikut:

1. Pada baris 5 adalah proses untuk mengubah semua karakter dalam dokumen menjadi karakter kecil.
2. Pada baris 6-7 adalah *regex* yang menyimpan pola URL.
3. Pada baris 8 adalah menghapus URL dari dokumen.
4. Pada baris 9 adalah hasil *case folding* yang disimpan pada variabel *doc*.

4.5.2 Implementasi *Data Cleaning*

Pada tahapan *cleaning* ini bertujuan untuk menghapus karakter tidak penting pada sebuah dokumen. Berikut ini adalah implementasi *cleaning* yang akan ditunjukkan pada Kode Program 4.19.

```
1   print "# data cleaning"
2   punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~0123456789'''
3   for kata in listkata:
4       no_punct = ""
5       for char in kata[1]:
6           if char not in punctuations:
7               no_punct = no_punct + char
8           elif char in punctuations:
9               no_punct = no_punct + ' '
10      kata[1] = no_punct
11      print kata[1]
```

Kode Program 4.19 Implementasi *Data Cleaning*

Penjelasan potongan Kode Program 4.19 tentang implementasi proses *cleaning* dijelaskan sebagai berikut:

1. Pada baris 3 adalah daftar tanda baca.
2. Pada baris 4-9 adalah pengecekan karakter pada daftar tanda baca. Jika karakter tidak ada pada daftar tanda baca, maka karakternya akan masuk variabel *no_punct*. Namun, jika karakter ada pada daftar tanda baca, maka karakternya akan dihapus.
3. Pada baris 10 adalah hasil *data cleaning* yang disimpan dalam variable *no_punct*.

4.5.3 Implementasi *Stemming*

Pada tahapan *stemming* ini bertujuan untuk mengubah kata menjadi kata dasar pada sebuah dokumen. Berikut ini adalah implementasi *stemming* yang akan ditunjukkan pada Kode Program 4.20.

```
1  print "# stemming"
2  stemmer = factory.create_stemmer()
3  for kata in listkata:
4      kata[1] = stemmer.stem(str(kata[1].lower()))
5      print kata[1]
```

Kode Program 4.20 Implementasi *Stemming*

Penjelasan potongan Kode Program 4.20 tentang implementasi proses *stemming* dijelaskan sebagai berikut:

1. Pada baris 2 adalah proses pembuatan *stemmer* dari *library* Sastrawi
2. Pada baris 4 adalah mengubah kata menjadi semua huruf kecil kemudian menghapus imbuhan pada kata.

4.5.4 Implementasi *Filtering*

Pada tahapan *filtering* ini bertujuan untuk memilih kata penting pada sebuah dokumen. Berikut ini adalah implementasi *filtering* yang akan ditunjukkan pada Kode Program 4.21.

```
1  print "# filtering"
2  stopword = open('stopword_list_tala.txt', 'r').read()
3  stopwordlist = re.split(r'\n', stopword)
4  for kolom in listkata:
5      wordstop = re.split(r'\s', kolom[1])
6      fil = ''
7      for y in wordstop:
8          if y not in stopwordlist:
9              fil += ' ' + y
10     kolom[1] = fil
11     print kolom[1]
```

Kode Program 4.21 Implementasi *Filtering*

Penjelasan potongan Kode Program 4.21 tentang implementasi proses *filtering* dijelaskan sebagai berikut:

1. Pada baris 2 adalah membuka dan membaca *file* stopword_list_tala.txt.
2. Pada baris 3 adalah memecah kata-kata dalam *stopword.*
3. Pada baris 5 adalah memecah kata-kata pada setiap dokumen.
4. Pada baris 7-10 adalah pengecekan kata pada *stopwordlist*. Jika kata tidak ada pada *stopwordlist* maka kata tersebut akan disimpan dalam variable *fil.*

Gambar 4.23 Cuplikan isi *file* stopword_list_tala.txt

4.5.5 Implementasi Tokenisasi

Pada tahapan tokenisasi ini bertujuan untuk memecah kata pada sebuah dokumen. Berikut ini adalah implementasi tokenisasi yang akan ditunjukkan pada Kode Program 4.22.

```
1   print "# tokenisasi"
2   for value in listkata:
3       baris = re.split(r'\s', value[1])
4       value[1] = baris
5
6   for dat in listkata:
7       for a in dat[1]:
8           if a =='':
9               dat[1].remove(a)
10      print dat
```

Kode Program 4.22 Implementasi Tokenisasi

Penjelasan potongan Kode Program 4.22 tentang implementasi proses tokenisasi dijelaskan sebagai berikut:

1. Pada baris 2-4 adalah memecah kata pada setiap dokumen.
2. Pada baris 6-9 adalah menghapus token yang kosong.

4.5.6 Implementasi *Raw Term Frequency*

Pada tahapan *raw term frequency* ini bertujuan untuk menghitung jumlah kata tertentu pada sebuah dokumen. Berikut ini adalah implementasi tokenisasi yang akan ditunjukkan pada Kode Program 4.23.

```
1   print "# kata unik"
2   unique = []
3   for token in listkata:
4       for unik in token[1]:
5           if unik not in unique:
6               unique.append(unik)
7       fiturkata = sorted(unique)
8   print "unik = ", unique
9
10  print "# term frequency"
11  doc_count = []
12  tf = []
13  for kata in fiturkata:
14      doc_count.append(0)
15      raw_term =[]
16      for f_doc in self.docs:
17          kt = f_doc[1].count(str(kata))
```

```
18              raw_term.append(kt)
19          tf.append(raw_term)
20          print "fitur  (",kata,") = ",raw_term
21      return fiturkata, tf
```

Kode Program 4.23 Implementasi *Raw Term Frequency*

Penjelasan potongan Kode Program 4.23 tentang implementasi proses *raw term frequency* dijelaskan sebagai berikut:

1. Pada baris 2 adalah inisialisasi *array* untuk menyimpan kata unik.
2. Pada baris 3-7 adalah membuat daftar kata unik setelah itu diurutkan dan dimasukkan ke dalam variabel fiturkata.
3. Pada baris 11-19 adalah menghitung *raw term frequency* dari setiap kata unik dan memasukkan nilai *raw term frequency* pada *array*.

4.6 Implementasi *Naïve Bayes*

Pada tahapan implementasi *Naïve Bayes*, terdapat beberapa tahapan yang dilakukan yaitu tahapan *Prior*, *Bernoulli Training*, *Bernoulli Testing*, *Gaussian*, *Multinomial Training*, *Multinomial Testing*, dan *Posterior*.

4.6.1 Implementasi *Prior*

Pada tahapan implementasi *prior* yang bertujuan untuk mencari peluang kemunculan sebuah kategori tertentu. Berikut ini adalah implementasi *prior* yang akan ditunjukkan pada Kode Program 4.24.

```
1   def Prior(self):
2       doc_pos = 0
3       doc_neg = 0
4       print "prior"
5       for doc in self.docslatih:
6           if doc[2].lower() == 'positif':
7               doc_pos += 1
8           else:
9               doc_neg += 1
10      prior_pos = float(doc_pos) / len(self.docslatih)
11      prior_neg = float(doc_neg) / len(self.docslatih)
12      self.prior_pos = prior_pos
13      self.prior_neg = prior_neg
14      print "prior_pos = ", prior_pos
15      print "prior_neg = ", prior_neg
16      return prior_pos, prior_neg
```

Kode Program 4.24 Implementasi *Prior*

Penjelasan potongan Kode Program 4.24 tentang implementasi proses prior dijelaskan sebagai berikut:

1. Pada baris 2-3 adalah inisialisasi awal dokumen kelas positif dan kelas negatif.
2. Pada baris 5-9 adalah menghitung jumlah dokumen latih kelas positif dan dokumen latih kelas negatif.
3. Pada baris 10 adalah menghitung *prior* dengan menghitung jumlah dokumen positif per jumlah dokumen data latih.
4. Pada baris 11 adalah menghitung *prior* dengan menghitung jumlah dokumen positif per jumlah dokumen data latih.

5. Pada baris 12 adalah memasukkan nilai prior positif ke variabel self.prior_pos.
6. Pada baris 13 adalah memasukkan nilai prior negatif ke variabel self.prior_pos.

4.6.2 Implementasi *Bernoulli Training*

Pada tahapan implementasi *Bernoulli Training* dilakukan proses yang bertujuan untuk menghitung nilai fitur yang representasi datanya biner pada dokumen latih. Berikut ini adalah implementasi *Bernoulli Training* yang akan ditunjukkan pada Kode Program 4.25.

```
1   def Bernoulli_training(self):
2       print "bernoulli training"
3       bfitur = [
4           self.fiturs[1],
5           self.fiturs[2],
6           self.fiturs[3],
7           self.fiturs[4],
8           self.fiturs[11],
9           self.fiturs[12],
10      ]
11      doc_pos = 0
12      doc_neg = 0
13      for doc in self.docslatih:
14          if doc[2].lower() == 'positif':
15              doc_pos += 1
16          else:
17              doc_neg += 1
18      tot_neg = []
19      tot_pos = []
20      for fitur in bfitur:
21          counter = 0
22          pos = 0
23          neg = 0
24          for doc in self.docslatih:
25              if doc[2].lower() == 'positif':
26                  pos += float(
27                      fitur[counter])
28              else:
29                  neg += float(fitur[counter])
30              counter += 1
31          tot_pos.append(pos)
32          tot_neg.append(neg)
33      pf_neg = []
34      for neg in tot_neg:
35          pf_neg.append((neg + 1) / (doc_neg + 2))
36      pf_pos = []
37      for pos in tot_pos:
38          pf_pos.append((pos + 1) / (doc_pos + 2))
39      self.bpf_pos = pf_pos
40      self.bpf_neg = pf_neg
41      print "pf_pos", pf_pos
42      print "pf_neg", pf_neg
43      return pf_pos, pf_neg
```

Kode Program 4.25 Implementasi *Bernoulli Training*

Penjelasan potongan Kode Program 5.25 tentang implementasi proses *Bernoulli Training* dijelaskan sebagai berikut:

1. Pada baris 3-10 adalah menyimpan nilai fitur dengan representasi data biner dari data latih.
2. Pada baris 13-17 adalah menghitung jumlah dokumen kelas positif dan kelas negatif.
3. Pada baris 18-19 adalah inisialisasi *array* untuk menyimpan nilai total dokumen kelas positif dan kelas negatif.
4. Pada baris 20-32 adalah menghitung setiap nilai fitur berdasarkan dokumen latih dan kelas.
5. Pada baris 34-38 adalah menghitung peluang fitur pada kelas positif dan kelas negatif berdasarkan dokumen latih.

4.6.3 Implementasi *Bernoulli Testing*

Pada tahapan implementasi *Bernoulli Testing* dilakukan proses yang bertujuan untuk menghitung nilai fitur yang representasi datanya biner pada dokumen uji. Berikut ini adalah implementasi *Bernoulli Testing* yang akan ditunjukkan pada Kode Program 4.26.

```
1   def Bernoulli_testing(self):
2       print "bernoulli testing"
3       bfitur = [
4           self.fiturs[1],
5           self.fiturs[2],
6           self.fiturs[3],
7           self.fiturs[4],
8           self.fiturs[11],
9           self.fiturs[12],
10      ]
11      prod_pos_b = []
12      prod_neg_b = []
13      for x in range(len(self.docslatih),len(self.docs)):
14          pd_pos = []
15          pd_neg = []
16          for i, fitur in enumerate(bfitur):
17              f_pd_neg = (self.bpf_neg[i] ** float(fitur[x])) * ((1
18  - self.bpf_neg[i]) ** (1 - float(fitur[x])))
19              f_pd_pos = (self.bpf_pos[i] ** float(fitur[x])) * ((1
20  - self.bpf_pos[i]) ** (1 - float(fitur[x])))
21              pd_pos.append(f_pd_pos)
22              pd_neg.append(f_pd_neg)
23          prod_pos_b.append(prod(pd_pos))
24          prod_neg_b.append(prod(pd_neg))
25      self.prod_pos_b = prod_pos_b
26      self.prod_neg_b = prod_neg_b
27      print "prod_pos_b", prod_pos_b
28      print "prod_neg_b", prod_neg_b
```

Kode Program 4.26 Implementasi *Bernoulli Testing*

Penjelasan potongan Kode Program 4.26 tentang implementasi proses *Bernoulli Testing* dijelaskan sebagai berikut:

1. Pada baris 3-10 adalah menyimpan nilai fitur dengan representasi data biner dari data uji.
2. Pada baris 11 adalah inisialisasi *array* untuk menyimpan nilai *likelihood* Bernoulli kelas positif.

3. Pada baris 12 adalah inisialisasi *array* untuk menyimpan nilai *likelihood* Bernoulli kelas negatif.
4. Pada baris 13-24 adalah menghitung peluang fitur pada kelas positif dan kelas negatif berdasarkan dokumen uji.
5. Pada baris 25-26 adalah menyimpan nilai *likelihood Bernoulli* kelas positif dan kelas negatif ke dalam variabel yang bisa diakses method lain.

4.6.4 Implementasi *Gaussian Naive Bayes*

Pada tahapan implementasi *Gaussian Naive Bayes* dilakukan proses yang bertujuan untuk menghitung nilai fitur yang representasi datanya kontinyu pada dokumen. Berikut ini adalah implementasi *Gaussian Naive Bayes* yang akan ditunjukkan pada Kode Program 4.27.

```
1   def Gaussian(self):
2       print "gaussian"
3       gfitur = [
4           self.fiturs[5],
5           self.fiturs[6],
6           self.fiturs[18],
7           self.fiturs[19],
8           self.fiturs[20],
9           self.fiturs[21],
10          self.fiturs[22],
11          self.fiturs[31],
12          self.fiturs[32],
13          self.fiturs[33],
14          self.fiturs[34],
15          self.fiturs[35],
16          self.fiturs[36],
17      ]
18      doc_pos = 0
19      doc_neg = 0
20      for doc in self.docslatih:
21          if doc[2].lower() == 'positif':
22              doc_pos += 1
23          else:
24              doc_neg += 1
25      rata2_neg = []
26      rata2_pos = []
27      i = 0
28      for fitur in gfitur:
29          counter = 0
30          pos = 0
31          neg = 0
32          for doc in self.docslatih:
33              if doc[2].lower() == 'positif':
34                  pos += float(
35                      fitur[counter])
36              else:
37                  neg += float(fitur[counter])
38              counter += 1
39          rata2_pos.append(pos / doc_pos)
40          rata2_neg.append(neg / doc_neg)
41      self.g_rata2_pos = rata2_pos
42      self.g_rata2_neg = rata2_neg
43      print "rata2_pos = ", rata2_pos
44      print "rata2_neg = ", rata2_neg
45
46      var_pos = []
```

```
47      var_neg = []
48      for i, fitur in enumerate(gfitur):
49          var_atas_pos = 0
50          var_atas_neg = 0
51          for j, doc in enumerate(self.docslatih):
52              if doc[2].lower() == 'positif':
53                  var_atas_pos += (float(fitur[j]) - rata2_pos[i])
54  ** 2
55              else:
56                  var_atas_neg += (float(fitur[j]) - rata2_neg[i])
57  ** 2
58          var_pos.append(var_atas_pos / doc_pos)
59          var_neg.append(var_atas_neg / doc_neg)
60      self.g_var_pos = var_pos
61      self.g_var_neg = var_neg
62      print "var_pos = ", var_pos
63      print "var_neg = ", var_neg
64
65      prod_pos_g = []
66      prod_neg_g = []
67      for x in range(len(self.docslatih), len(self.docs)):
68          p_pd_g_pos = []
69          p_pd_g_neg = []
70          for i, fitur in enumerate(gfitur):
71              if self.g_var_pos[i] == 0:
72                  g_pembilang_pos = 1
73                  g_penyebut_pos = 1
74              else:
75                  g_pembilang_pos = ((1 / math.sqrt(2 * 3.14 *
76  self.g_var_pos[i])))
77                  g_penyebut_pos = (2.17 ** (-((float(fitur[x]) -
78  self.g_rata2_pos[i]) ** 2) / (2 * self.g_var_pos[i])))
79              if self.g_var_neg[i] == 0:
80                  g_pembilang_neg = 1
81                  g_penyebut_neg = 1
82              else:
83                  g_pembilang_neg = ((1 / math.sqrt(2 * 3.14 *
84  self.g_var_neg[i])))
85                  g_penyebut_neg = (2.17 ** (-((float(fitur[x]) -
86  self.g_rata2_neg[i]) ** 2) / (2 * self.g_var_neg[i])))
87              p_pd_g_pos.append(g_pembilang_pos * g_penyebut_pos)
88              p_pd_g_neg.append(g_pembilang_neg * g_penyebut_neg)
89          prod_pos_g.append(prod(p_pd_g_pos))
90          prod_neg_g.append(prod(p_pd_g_neg))
91      self.prod_pos_g = prod_pos_g
92      self.prod_neg_g = prod_neg_g
93      print "prod_pos_g ", prod_pos_g
94      print "prod_neg_g ", prod_neg_g
```

Kode Program 4.27 Implementasi *Gaussian Naive Bayes*

Penjelasan potongan Kode Program 4.27 tentang implementasi proses Gaussian dijelaskan sebagai berikut:

1. Pada baris 3-17 adalah menyimpan nilai fitur dengan representasi data kontinyu pada *array*.
2. Pada baris 18-24 adalah menghitung jumlah dokumen kelas positif dan kelas negatif.
3. Pada baris 28-38 adalah menghitung nilai total dokumen kelas positif dan kelas negatif kemudian disimpan ke dalam *array*.
4. Pada baris 39-40 adalah menghitung nilai rata-rata kelas positif dan kelas negatif.

5. Pada baris 46-59 adalah menghitung nilai variansi kelas positif dan kelas negatif.
6. Pada baris 67-88 adalah menghitung nilai *likelihood Gaussian* kelas positif dan kelas negatif.
7. Pada baris 89-90 adalah memasukkan nilai *likelihood Gaussian* ke dalam *array*.
8. Pada baris 91-92 adalah mengakses nilai *likelihood Gaussian* kelas positif dan kelas negatif.

4.6.5 Implementasi *Multinomial Training*

Pada tahapan implementasi *Multinomial Training* dilakukan proses yang bertujuan untuk menghitung nilai fitur yang representasi datanya diskrit pada dokumen latih. Berikut ini adalah implementasi *Multinomial Training* yang akan ditunjukkan pada Kode Program 4.28.

```
1   def Multinomial_training(self):
2       print "multinomial training"
3       mfitur = [
4           self.fiturs[7],
5           self.fiturs[8],
6           self.fiturs[9],
7           self.fiturs[10],
8           self.fiturs[13],
9           self.fiturs[14],
10          self.fiturs[15],
11          self.fiturs[16],
12          self.fiturs[17],
13          self.fiturs[23],
14          self.fiturs[24],
15          self.fiturs[25],
16          self.fiturs[26],
17          self.fiturs[27],
18          self.fiturs[28],
19          self.fiturs[29],
20          self.fiturs[30],
21          self.fiturs[37],
22      ]
23      if len(self.fiturs)>37 :
24          for i in range(38, len(self.fiturs)):
25              mfitur.append(self.fiturs[i])
26      doc_pos = 0
27      doc_neg = 0
28      for doc in self.docslatih:
29          print doc
30          if doc[2].lower() == 'positif':
31              doc_pos += 1
32          else:
33              doc_neg += 1
34          tot_neg = []
35          tot_pos = []
36          i = 0
37          for fitur in mfitur:
38              counter = 0
39              pos = 0
40              neg = 0
41              for doc in self.docs:
42                  if doc[2].lower() == 'positif':
43                      pos += float(fitur[
```

```
44                                          counter])
45                  else:
46                      neg += float(fitur[counter])
47                  counter += 1
48              tot_pos.append(pos)
49              tot_neg.append(neg)
50              i += 1
51              print "fitur ",i
52          pf_neg = []
53          for neg in tot_neg:
54              pf_neg.append(round((neg + 1) / (sum(tot_neg) +
55  len(tot_neg)), 4))
56          self.m_pf_neg = pf_neg
57          pf_pos = []
58          for pos in tot_pos:
59              pf_pos.append(round((pos + 1) / (sum(tot_pos) +
60  len(tot_pos)), 4))
61          self.m_pf_pos = pf_pos
62      print "pf_pos = ", (pf_pos)
63      print "pf_neg = ", (pf_neg)
```

Kode Program 4.28 Implementasi Multinominal Training

Penjelasan potongan Kode Program 4.28 tentang implementasi proses Multinomial Training dijelaskan sebagai berikut:

1. Pada baris 3-22 adalah menyimpan nilai fitur dengan representasi data diskrit pada *array*.
2. Pada baris 23-25 adalah mengambil fitur *bag of words* dari *file* csv untuk dimasukkan ke dalam *array* mfitur.
3. Pada baris 26-33 adalah menghitung jumlah dokumen kelas positif dan kelas negatif.
4. Pada baris 34-50 adalah menyimpan nilai total dokumen kelas positif dan kelas negatif pada *array*.
5. Pada baris 52-61 adalah menghitung peluang fitur pada kelas positif dan kelas negatif berdasarkan dokumen latih.

4.6.6 Implementasi *Multinomial Testing*

Pada tahapan implementasi *Multinomial Testing* dilakukan proses yang bertujuan untuk menghitung nilai fitur yang representasi datanya diskrit pada dokumen uji. Berikut ini adalah implementasi *Multinomial Testing* yang akan ditunjukkan pada Kode Program 4.29.

```
1   def Multinomial_testing(self):
2       print "Multinomial testing"
3       mfitur = [
4           self.fiturs[7],
5           self.fiturs[8],
6           self.fiturs[9],
7           self.fiturs[10],
8           self.fiturs[13],
9           self.fiturs[14],
10          self.fiturs[15],
11          self.fiturs[16],
12          self.fiturs[17],
13          self.fiturs[23],
14          self.fiturs[24],
15          self.fiturs[25],
```

```
            self.fiturs[26],
            self.fiturs[27],
            self.fiturs[28],
            self.fiturs[29],
            self.fiturs[30],
            self.fiturs[37],
        ]
        if len(self.fiturs)>37 :
            for i in range(38, len(self.fiturs)):
                mfitur.append(self.fiturs[i])
        prod_pos_m = []
        prod_neg_m = []
        for x in range(len(self.docslatih), len(self.docs)):
            print self.docs[x]
            pd_pos = []
            pd_neg = []
            for i, fitur in enumerate(mfitur):
                f_pd_pos = (self.m_pf_pos[i] ** float(fitur[x]))
                f_pd_neg = (self.m_pf_neg[i] ** float(fitur[x]))
                pd_pos.append(f_pd_pos)
                pd_neg.append(f_pd_neg)
            prod_pos_m.append(prod(pd_pos))
            prod_neg_m.append(prod(pd_neg))
        self.prod_pos_m = prod_pos_m
        self.prod_neg_m = prod_neg_m
        print "prod_pos_m = ", prod_pos_m
        print "prod_neg_m = ", prod_neg_m
```

Kode Program 4.29 Implementassi *Multinominal Testing*

Penjelasan potongan Kode Program 4.29 tentang implementasi proses Multinomial Testing dijelaskan sebagai berikut:

1. Pada baris 3-22 adalah menyimpan nilai fitur dengan representasi data diskrit pada *array*.
2. Pada baris 23-25 adalah mengambil fitur *bag of words* dari *file* csv untuk dimasukkan ke dalam *array* mfitur.
3. Pada baris 28-38 adalah menghitung peluang fitur pada kelas positif dan kelas negatif berdasarkan dokumen uji dan memasukkan nilai tersebut ke dalam *array*.
4. Pada baris 39-40 adalah menyimpan nilai *likelihood* Multinomial kelas positif dan kelas negative ke dalam variabel yang bisa diakses method lain.

4.6.7 Implementasi *Posterior*

Pada tahapan implementasi *posterior* yang bertujuan untuk mencari peluang kategori sesuatu ketika terdapat kemunculan kata tertentu. Berikut ini adalah implementasi *posterior* yang akan ditunjukkan pada Kode Program 4.30.

```
def Posterior(self):
    print "posterior"
    result = []
    juji = len(self.docs)-self.jlatih
    for x in range(0,juji):
        nilai_posterior_pos = self.prior_pos * self.prod_pos_b[x]
* self.prod_pos_g[x] * self.prod_pos_m[x]
        self.nilai_posterior_pos = nilai_posterior_pos
        print "nilai_posterior_pos = ", nilai_posterior_pos

        nilai_posterior_neg = self.prior_neg * self.prod_neg_b[x]
* self.prod_neg_g[x] * self.prod_neg_m[x]
```

```
13          self.nilai_posterior_neg = nilai_posterior_neg
14          print "nilai_posterior_neg = ", nilai_posterior_neg
15
16          if self.nilai_posterior_pos >= self.nilai_posterior_neg:
17              result.append("POSITIF")
18          else:
19              result.append("NEGATIF")
20      print result
21      return result
```

Kode Program 4.30 Implementasi *Posterior*

Penjelasan potongan Kode Program 4.30 tentang implementasi proses *Posterior* dijelaskan sebagai berikut:

1. Pada baris 4 adalah menghitung jumlah data uji dengan melakukan pengurangan antara jumlah total dokumen dan jumlah data latih.
2. Pada baris 5-14 adalah menghitung nilai *posterior* kelas positif dan kelas negatif dengan mengkalikan nilai *prior* dengan *likelihood* Bernoulli, Gaussian, dan Multinomial per data uji.
3. Pada baris 16-19 adalah jika nilai *posterior* positif lebih besar sama dengan nilai *posterior* negatif maka hasilnya akan menjadi "POSITIF" jika tidak maka hasilnya "NEGATIF".

4.7 Implementasi Pengujian

Pada tahapan ini akan dilakukan proses implementasi pengujian yang bertujuan untuk mengukur keakuratan suatu metode dalam menyelesaikan masalah seperti analisis sentimen. Berikut ini adalah beberapa tahapan pengujian yaitu perhitungan *confusion matrix*, *accuracy, precision, recall,* dan *f-measure*.

4.7.1 Implementasi Perhitungan *Confusion Matrix*.

Pada tahapan ini akan dilakukan proses perhitungan *confusion matrix* yang bertujuan untuk menghitung berapa jumlah data uji dengan hasil klasifikasi benar dan salah. Berikut ini adalah implementasi perhitungan *confusion matrix* yang akan ditunjukkan pada Kode Program 4.31.

```
1   def confusion(self):
2       tp = tn = fp = fn = 0
3       for a in range(0,len(self.classreal)):
4           if self.classreal[a][2].lower() ==
5   self.classresult[a].lower():
6               if self.classreal[a][2].lower() == 'positif':
7                   tp += 1
8               else:
9                   tn += 1
10          else:
11              if self.classreal[a][2].lower() == 'positif':
12                  fn += 1
13              else:
14                  fp += 1
15      self.tp = tp
16      self.tn = tn
17      self.fp = fp
18      self.fn = fn
19      print "tp =", tp
20      print "tn =", tn
```

```
21      print "fp =", fp
22      print "fn =", fn
```

Kode Program 4.31 Implementasi Perhitungan *Confusion Matrix*

Penjelasan potongan Kode Program 4.31 tentang implementasi proses *confusion matrix* dijelaskan sebagai berikut:

1. Pada baris 3-13 adalah proses pencocokan antara hasil analisis sentimen program dan hasil klasifikasi sebenarnya pada data uji. Setiap data uji yang mempunyai hasil pencocokan dan bersentimen positif, maka akan dihitung jumlahnya dalam variabel *tp* (*true positive*) dan jika bersentimen negatif maka akan dihitung jumlahnya dalam variabel *tn* (*true negative*). Apabila setiap data uji yang mempunyai hasil pencocokan salah dan seharusnya bersentimen positif, maka akan dihitung jumlahnya dalam variabel *fn* (*false negative*) dan jika seharusnya bersentimen negatif, maka akan dihitung jumlahnya dalam variabel *fp* (*false positive*).

4.7.2 Implementasi Perhitungan *Accuracy*

Pada tahapan ini akan dilakukan proses perhitungan *accuracy* yang bertujuan untuk mengetahui kesesuaian nilai hasil prediksi pengujian dengan nilai aktual yang dibandingkan. Berikut ini adalah implementasi *accuracy* yang akan ditunjukkan pada Kode Program 4.32.

```
1   def accuracy(self):
2       self.accuracy =
3   float(self.tn+self.tp)/float(self.tn+self.tp+self.fp+self.fn)
4       print "accuracy = ",(self.accuracy*100),"%"
```

Kode Program 4.32 Implementasi Perhitungan *Accuracy*

Penjelasan potongan Kode Program 4.32 tentang implementasi proses *accuracy* dijelaskan sebagai berikut:

1. Pada baris 2-3 adalah menghitung akurasi dengan membagi antara hasil klasifikasi yang benar dengan seluruh hasil klasifikasi data uji.

4.7.3 Implementasi Perhitungan *Precision*

Pada tahapan ini akan dilakukan proses perhitungan *precision* yang bertujuan untuk mengetahui tingkat ketepatan antara informasi yang diminta oleh *user* dengan hasil jawaban yang diberikan oleh sistem. Berikut ini adalah implementasi *precision* yang akan ditunjukkan pada Kode Program 4.33.

```
1   def precision(self):
2       self.precision = float(self.tp)/float(self.tp+self.fp)
3       print "precision = ", (self.precision)
```

Kode Program 4.33 Implementasi Perhitungan *Precision*

Penjelasan potongan Kode Program 4.33 tentang implementasi proses *precision* dijelaskan sebagai berikut:

1. Pada baris 2 adalah proses perhitungan *precision* dengan menghitung jumlah hasil klasifikasi bersentimen positif yang benar dibagi seluruh hasil klasifikasi sistem yang bersentimen positif.

4.7.4 Implementasi Perhitungan *Recall*

Pada tahapan ini akan dilakukan proses perhitungan *recall* yang bertujuan untuk mengetahui tingkat jumlah banyak dan sedikitnya kesesuaian informasi yang didapatkan dari hasil percobaan berdasarkan sudut pandang kelas atau label yang digunakan. Berikut ini adalah implementasi *recall* yang akan ditunjukkan pada Kode Program 4.34.

```
1   def recall(self):
2       self.recall = float(self.tp)/float(self.tp+self.fn)
3       print "recall = ", (self.recall)
```

Kode Program 4.34 Implementasi Perhitungan *Recall*

Penjelasan potongan Kode Program 4.34 tentang implementasi proses *recall* dijelaskan sebagai berikut:

1. Pada baris 2 adalah proses perhitungan *recall* dengan menghitung jumlah hasil klasifikasi manual atau yang sebenarnya bersentimen positif.

4.7.5 Implementasi Perhitungan *F-Measure*

Pada tahapan ini akan dilakukan proses perhitungan *f-measure* yang bertujuan untuk mengetahui bobot *harmonic mean* pada *recall* dan *precision*. Berikut ini adalah implementasi *f-measure* yang akan ditunjukkan pada Kode Program 4.35.

```
1   def fmeasure(self):
2       fmeasure =
3   (2*self.precision*self.recall)/(self.precision+self.recall)
4       print "F-Measure = ", (fmeasure)
```

Kode Program 4.35 Implementasi Perhitungan *F-Measure*

Penjelasan potongan Kode Program 4.35 tentang implementasi proses *f-measure* dijelaskan sebagai berikut:

1. Pada baris 2-3 adalah proses perhitungan *f-measure*.

4.8 Tampilan Program Sistem

Pada subbab ini menjelaskan tentang tampilan program sistem yang dimulai dari tampilan program input satu data uji beserta kelasnya yang sebenarnya, pembentukan fitur berupa *ensemble features* dan *bag of words features*, perhitungan Naive Bayes dan pengujian.

4.8.1 Tampilan Program Input Data Uji

Tampilan program di bawah ini adalah tampilan untuk menginputkan satu data yang akan diuji di dalam sistem klasifikasi beserta kelasnya yang sebenarnya agar bisa dibandingkan dengan kelas hasil klasifikasi. Tampilan program input data uji ditunjukkan pada Gambar 4.24.

```
============================================================
ANALISIS SENTIMEN FILM PADA TWITTER
============================================================
Masukkan data yang ingin diuji : Tapi Film yang aku kasih bagus kan mas io RT @RioRani0 After earth gak bagus ah film nya
Masukkan kelas berdasarkan data uji tersebut : Negatif
```

Gambar 4.24 Tampilan Program Input Data Uji

4.8.2 Tampilan Program Proses Pembuatan *Ensemble Features* dan *Bag of Word Features*

Tampilan program di bawah ini adalah tampilan untuk proses pembuatan *ensemble features* dan *bag of words features* yang berguna untuk perhitungan klasifikasi Naive Bayes pada proses selanjutnya. Tampilan program proses pembuatan *ensemble features* dan *bag of words features* ditunjukkan pada Gambar 4.25.

```
# Membuat Ensemble Features + Bag of Words
F1
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
F2
[0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
F3
[1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
F4
[1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
F5
[13, 19, 24, 13, 14, 19, 18, 30, 16, 26, 11, 15, 20, 16, 18, 6, 15, 21, 15, 16, 17, 21, 20, 12, 12, 38, 15, 22, 29, 13, 25, 12, 24,
F6
[14.0, 6.6667, 5.6667, 8.0, 7.6471, 5.0476, 11.381, 6.5882, 7.8889, 8.8462, 6.7273, 6.9412, 6.3913, 6.2353, 7.2105, 5.5714, 5.5294,
F7
[0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
F8
[0, 1, 0, 6, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
F9
[1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
F10
[10, 10, 2, 6, 14, 3, 14, 9, 3, 2, 6, 7, 5, 2, 6, 1, 5, 0, 4, 4, 2, 7, 9, 1, 11, 20, 14, 2, 7, 12, 4, 8, 0, 2, 2, 2, 2, 4, 3, 2, 5,
F11
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Gambar 4.25 Tampilan Program Proses Pembuatan *Ensemble Features* dan *Bag of Word Features*

4.8.3 Tampilan Program Perhitungan Naive Bayes

Tampilan program di bawah ini adalah tampilan untuk proses perhitungan Naive Bayes untuk menentukan hasil klasifikasi data uji. Tampilan program perhitungan Naive Bayes beserta kelas hasil klasifikasi data uji ditunjukkan pada Gambar 4.26.

```
# Perhitungan Naive Bayes
bernoulli training
pf_pos [0.1977401129943503, 0.1638418079096045, 0.4632768361581921, 0.1864406779661017, 0.0847457627118644, 0.5649717514124294]
pf_neg [0.24293785310734464, 0.12994350282485875, 0.4011299435028249, 0.14689265536723164, 0.05084745762711865, 0.5819209039548022]
bernoulli testing
prod_pos_b [0.019725536691438464]
prod_neg_b [0.013358858006276658]
gaussian
prod_pos_g  [924.7623036899696]
prod_neg_g  [1070.8871164387422]
multinomial training
pf_pos =  [0.0101, 0.0108, 0.0038, 0.1255, 0.0882, 0.0364, 0.0358, 0.0197, 0.0032, 0.0324, 0.0161, 0.0155, 0.0044, 0.0019, 0.0031, 0
pf_neg =  [0.0118, 0.0045, 0.0037, 0.1209, 0.085, 0.0311, 0.0374, 0.0252, 0.0034, 0.0163, 0.0263, 0.007, 0.0062, 0.0016, 0.0034, 0.0
prod_pos_m =  [9.081011113994465e-57]
prod_neg_m =  [5.822221582277472e-56]
prior
prior_pos =  0.5
prior_neg =  0.5
Hasil Klasifikasi =  ['NEGATIF']
```

Gambar 4.26 Tampilan Program Perhitungan Naive Bayes

4.8.4 Tampilan Program Pengujian

Tampilan program di bawah ini adalah tampilan untuk proses pengujian yang meliputi perhitungan akurasi untuk mengetahui apakah sistem klasifikasi dengan baik atau tidak. Tampilan program pengujian ditunjukkan pada Gambar 4.27.

```
# Pengujian
tp = 0
tn = 1
fp = 0
fn = 0
accuracy =  100.0 %
```

Gambar 4.27 Tampilan Program Pengujian

BAB 5 HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan mengenai pengujian pada sistem serta analisis terhadap hasil pengujian yang telah dilakukan. Untuk pengujian ini menggunakan 350 data latih yang terdiri dari 175 sentimen positif dan 175 sentimen negatif, sedangkan 150 data uji yang yang terdiri dari 75 sentimen positif dan 75 sentimen negatif. Berikut ini adalah skenario pengujian yang dilakukan meliputi pengaruh *Twitter specific features*, pengujian pengaruh *textual features*, pengujian pengaruh *part of speech features*, pengujian pengaruh *lexicon based features*, pengujian pengaruh *bag of words features*, pengujian pengaruh *ensemble features*, dan pengujian pengaruh *ensemble features* dan *bag of words features*.

5.1 Pengujian Pengaruh *Twitter Specific Features*

Pada pengujian ini, fitur yang digunakan hanya ada 4 meliputi apakah *tweet* mengandung *hashtag*, *retweet*, *username*, dan URL. Tujuan dari pengujian ini yaitu untuk mengetahui apakah *Twitter specific features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *Twitter specific features* yang akan ditunjukkan pada Tabel 5.1.

Tabel 5.1 Hasil Pengujian Twitter *Specific Features*

Twitter Specific features			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	29	43
	Negatif	46	32
Accuracy		40,67 %	
Precision		0,4028	
Recall		0,3867	
F-measure		0,3946	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *Twitter specific features* mempunyai pengaruh pada analisis sentimen ini. Namun, hanya menghasilkan akurasi sebesar 40,67% karena hanya terdapat 4 fitur dan fitur-fitur tersebut belum sepenuhnya dapat mewakili kelas dari suatu data atau *tweet*. *Twitter specific features* tersebar pada data latih kelas positif dan negatif, sehingga tidak merujuk ke suatu kelas tertentu dengan pasti. Selain itu, tidak semua *tweet* mengandung *Twitter specific features*. Hal itu menyebabkan sistem mengeluarkan hasil akurasi yang rendah. Karena itu juga, nilai *precision*-nya hanya 0,4028 dan *recall*-nya hanya 0,3868 yang berarti analisis sentimen yang

menggunakan fitur ini memiliki ketepatan klasifikasi yang rendah dan sedikitnya kesesuaian informasi yang didapat dari percobaan serta dari data yang sebenarnya. Sehingga nilai *f-measure* yang didapatkan hanya 0,3946.

5.2 Pengujian Pengaruh *Textual Features*

Pada pengujian ini, fitur yang digunakan ada 8 meliputi panjang *tweet*, rata-rata panjang kata, jumlah tanda tanya, jumlah tanda seru, jumlah tanda kutip, jumlah kata yang diawali huruf besar serta apakah *tweet* mengandung *emoticon* positif dan *emoticon* negatif. Tujuan dari pengujian ini yaitu untuk mengetahui apakah *Textual features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *Textual features* yang akan ditunjukkan pada Tabel 5.2.

Tabel 5.2 Hasil Pengujian *Textual Features*

Textual features			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	28	12
	Negatif	47	63
Accuracy		60,67 %	
Precision		0,7	
Recall		0,3733	
F-measure		0,4870	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *Textual features* mempunyai pengaruh pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 60,67% karena terdapat 8 fitur yang dapat mewakili sebagian kelas dari suatu data atau *tweet* contohnya seperti *emoticon* positif dan negatif. Namun, tidak semua *Textual Features* dapat mewakili kelas tertentu contohnya fitur tanda seru yang dapat ditemukan di data positif dan negatif. Selain itu, nilai *precision*-nya sebesar 0,7 yang menunjukkan penggunaan fitur ini pada analisis sentimen memiliki ketepatan yang cukup tinggi. Sementara *recall*-nya hanya 0,3733 yang berarti analisis sentimen yang menggunakan fitur ini memiliki sedikit kesesuaian informasi yang didapat dari percobaan serta dari data yang sebenarnya. Sehingga nilai *f-measure* yang didapatkan hanya 0,4870.

5.3 Pengujian Pengaruh *Part of Speech Features*

Pada pengujian ini, terdapat 10 fitur yang digunakan dalam analisis sentimen meliputi jumlah kata benda, sifat, kerja, keterangan, seru dan persentase kata benda, sifat, kerja, keterangan, seru. Tujuan dari pengujian ini yaitu untuk mengetahui apakah *Part of Speech features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *Part of Speech features* yang akan ditunjukkan pada Tabel 5.3.

Tabel 5.3 Hasil Pengujian *Part of Speech Features*

Part of Speech features			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	56	44
	Negatif	19	31
Accuracy		58,0 %	
Precision		0,56	
Recall		0,7467	
F-measure		0,64	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *Part of Speech features* mempunyai pengaruh pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 58,0% karena memiliki 10 fitur yang belum sepenuhnya dapat mewakili kelas dari suatu datau atau tweet. Fitur-fitur yang terdiri dari jumlah kata benda, sifat, kerja, keterangan, seru dan persentase kata benda, sifat, kerja, keterangan, seru belum dapat menentukan suatu data dapat masuk ke kelas positif atau negatif dengan tepat. Banyaknya jumlah kata pada suatu data tidak bisa menjadi rujukan untuk menentukan kelas suatu data dengan pasti. Seperti apakah bila suatu data yang memiliki banyak kata benda tidak selalu akan masuk ke kelas negatif atau positif. Begitupun dengan kata sifat, kerja, keterangan, seru, dan persentase jumlah dari kata-kata tersebut. *Part of speech features* hanya mengecek kata-kata baku saja dan tidak memperhatikan kata-kata tidak baku sehingga tidak semua fitur kata terhitung dalam *Part of speech features* dan menyebabkan kesalahan hasil klasifikasi. sementara bag of words memperhatikan kata-kata tidak baku juga dalam menghitung fitur kata sehingga seluruh kata dianggap penting dan terhitung.

Selain itu, nilai *precision*-nya sebesar 0,56 yang berarti analisis sentimen yang menggunakan fitur ini memiliki ketepatan yang sedang. Sementara *recall*-nya sebesar 0,7467 yang berarti analisis sentimen yang menggunakan fitur ini memiliki banyak kesesuaian informasi yang didapat dari percobaan serta dari data yang sebenarnya. Sehingga nilai *f-measure* yang didapatkan sebesar 0,64.

5.4 Pengujian Pengaruh *Lexicon Based Features*

Pada pengujian ini, terdapat 15 fitur yang digunakan dalam sistem ini meliputi jumlah kata positif dan negatif, kata sifat yang bersentimen positif dan negatif, kata kerja yang bersentimen positif dan negatif, kata keterangan yang bersentimen positif dan negatif, persentase kata sifat yang bersentimen positif dan negatif, kata kerja yang bersentimen positif dan negatif, kata keterangan yang bersentimen positif dan negatif serta jumlah kata *intensifier*. Tujuan dari pengujian ini yaitu untuk mengetahui apakah *Lexicon Based features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *Lexicon based features* yang akan ditunjukkan pada Tabel 5.4.

Tabel 5.4 Hasil Pengujian *Lexicon based Features*

Lexicon based features			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	38	20
	Negatif	37	55
Accuracy		62,0 %	
Precision		0,6552	
Recall		0,5067	
F-measure		0,5714	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *lexicon based features* mempunyai pengaruh pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 62,0% karena terdapat 15 fitur yang dapat mewakili sebagian kelas dari suatu data atau *tweet* seperti fitur jumlah kata positif yang dapat menunjukkan suatu data uji masuk ke kelas positif dan sebaliknya. Namun, *lexicon based features* juga dapat membuat analisis sentimen kurang tepat dalam klasifikasi karena ada kondisi yang mana kata positif terdapat dalam data uji kelas negatif begitupun sebaliknya, namun ketika data tersebut masuk dalam sistem malah terdeteksi sebagai data positif. Hal ini juga menyebabkan nilai *precision*-nya sebesar 0,6552. Sementara untuk nilai *recall*-nya sebesar 0,5067 yang berarti analisis sentimen yang menggunakan fitur ini memiliki kesesuaian informasi yang cukup yang didapat dari percobaan serta dari data yang sebenarnya. Sehingga nilai *f-measure* yang didapatkan sebesar 0,5714.

5.5 Pengujian Pengaruh *Bag of Words Features*

Pada pengujian ini, fitur yang digunakan adalah *bag of words features* yaitu fitur dari sisi statistik kata yang memiliki tujuan untuk mengetahui apakah *bag of words features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *bag of words features* yang ditunjukkan pada Tabel 5.5.

Tabel 5.5 Hasil Pengujian *Bag of Words Features*

Bag of Words features			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	69	3
	Negatif	6	72
Accuracy		94,0 %	
Precision		0,9583	
Recall		0,92	
F-measure		0,9388	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *bag of words features* mempunyai pengaruh pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 94,0 % karena fitur statistik kata yang dapat mewakili kelas dari suatu data atau *tweet*. Setiap kata hasil dari proses *case folding*, *data cleaning*, *stemming, filtering* dan tokenisasi dari data latih maupun data uji akan menjadi fitur *bag of words* dengan nilai sebanyak munculnya kata tersebut di setiap dokumen. Fitur kata pada bag of words tergantung dari data latihnya, tidak seperti *lexicon based features* yang tergantung pada daftar kata bersentimennya atau part of speech yang tergantung pada Kamus Besar Bahasa Indonesia. Kata-kata yang tidak termasuk ke dalam daftar kata bersentimen atau Kamus Besar Bahasa Indonesia (yang biasa disebut kata tidak baku) akan tetap tercatat sebagai *bag of words features* dan dihitung jumlah frekuensinya dari setiap dokumen. Fitur kata yang sering muncul di data latih dengan kelas negatif akan memiliki nilai frekuensi yang tinggi untuk kelas negatif juga, sehingga fitur kata tersebut dapat menjadi wakil atau petunjuk dalam proses klasifikasi data ujinya untuk menunjukkan mana data uji yang akan masuk kelas negatif dan sebaliknya. Sebagai contoh, seperti kata 'tidak' yang sering muncul di data yang memiliki kelas negatif. Semakin banyak jumlah data latih nya maka fitur katanya akan semakin beragam dan akan menghasilkan banyak keluaran klasifikasi yang tepat sehingga membantu meningkatkan akurasi proses klasifikasi.

Nilai *precision*-nya juga tinggi yaitu sebesar 0,9583 yang berarti analisis sentimen yang menggunakan fitur ini memiliki ketepatan yang tinggi. Sementara

untuk nilai *recall*-nya sebesar 0,92 yang berarti analisis sentimen yang menggunakan fitur ini memiliki banyak kesesuaian informasi yang didapat dari percobaan serta dari data yang sebenarnya maka akan didapatkan nilai *f-measure* yang didapatkan sebesar 0,9388.

5.6 Pengujian Pengaruh *Ensemble Features* tanpa *Bag of Words*

Pada pengujian ini, fitur yang digunakan adalah *ensemble features* yaitu gabungan dari beberapa fitur dari sisi semantik kata. Tujuannya untuk mengetahui pengaruh *ensemble features* pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *ensemble features* tanpa *bag of words* yang akan ditunjukkan pada Tabel 5.6.

Tabel 5.6 Hasil Pengujian *Ensemble Features* tanpa *Bag of Words*

Ensemble features tanpa *Bag of words*			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	41	24
	Negatif	34	51
Accuracy		61,33 %	
Precision		0,6308	
Recall		0,5467	
F-measure		0,5857	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *ensemble features* mempunyai pengaruh pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 61,33 % karena semua fitur mulai dari *Twitter specific features*, *textual features*, *part of speech features*, dan *lexicon based features* menjadi satu. Hal ini disebabkan karena fitur-fitur tersebut dapat mewakili kelas dari suatu data atau *tweet*. Namun, adakalanya fitur-fitur tersebut tidak bisa mewakili kelas dengan benar sehingga menyebabkan akurasinya menjadi seperti itu. Misalnya *ensemble features* yang bagian *lexicon based features*. Pada *lexicon based features*, fitur yang digunakan untuk mewakili suatu kelas hanyalah fitur kata positif dan negatif yang terdaftar pada daftar kata bersentimen atau disebut dengan *sentiwords* seperti kata 'bagus' yang termasuk pada kata bersentimen positif. Namun kata 'bagus' tersebut tak selalu hanya berada di data yang mempunyai kelas positif saja, tapi juga terdapat pada data dengan kelas negatif seperti pada kalimat "Tapi Film yang aku kasih bagus kan mas io RT @RioRani0 After earth gak bagus ah film nya". Karena *lexicon based features* hanya mendeteksi kata-kata bersentimen yang masuk dalam *sentiwords* saja, sehingga data yang seharusnya masuk ke dalam kelas negatif tersebut menjadi masuk ke kelas positif. Demikian juga dengan Twitter *features*, *Textual features*, dan *Part*

of speech features yang fiturnya masih belum mewakili suatu kelas dengan pasti atau hanya sedikit dari fiturnya yang dapat mewakili suatu kelas jarang ditemukannya fitur tersebut pada kebanyakan data. Inilah penyebab dari bentuk kesalahan dalam hasil klasifikasi pada sistem analisis sentimen yang hanya menggunakan *ensemble features* saja di dalamnya dan menyebabkan hasil akurasinya lebih rendah dari *bag of words features*.

Sementara itu, nilai *precision*-nya sebesar 0,6308 dan nilai *recall*-nya sebesar 0,5467 yang berarti tingkat ketepatan analisis sentimen ini cukup bagus dan terdapat kesesuaian informasi yang cukup dan didapat dari percobaan serta dari data yang sebenarnya maka akan didapatkan nilai *f-measure* yang didapatkan sebesar 0,5857.

5.7 Pengujian Pengaruh *Ensemble Features* dan *Bag of Words Features*

Pada pengujian ini, fitur yang digunakan adalah *ensemble features* dan *bag of words features* yang memiliki tujuan untuk mengetahui *ensemble features* berpengaruh pada tingkat akurasi dari analisis sentimen ini. Berikut ini adalah hasil pengujian pengaruh *ensemble features* dan *bag of words* yang ditunjukkan pada Tabel 5.7.

Tabel 5.7 Hasil Pengujian *Ensemble Features dan Bag of Words Features*

Ensemble features + bag of words			
		Hasil sebenarnya	
		Positif	Negatif
Hasil Prediksi	Positif	64	6
	Negatif	11	69
Accuracy		88,67 %	
Precision		0,9143	
Recall		0,8533	
F-measure		0,8828	

Berdasarkan hasil pengujian di atas, dapat diketahui bahwa *ensemble features* dan *bag of words features* yang terdiri dari *Twitter specific features*, *textual features*, *part of speech features*, *lexicon based features* dan *bag of words features* menjadi satu tersebut mempunyai pengaruh yang baik pada analisis sentimen ini yaitu menghasilkan akurasi sebesar 88.67 %. Hal ini disebabkan karena fitur-fitur tersebut dapat mewakili kelas dari suatu data atau *tweet*. Seluruh kelebihan dari fitur tersebut tergabung dan saling melengkapi sehingga menghasilkan banyak hasil klasifikasi yang keluar dengan benar. Setiap fitur dari ensemble fetures mempunyai kelebihan dan kekurangan tersendiri seperti

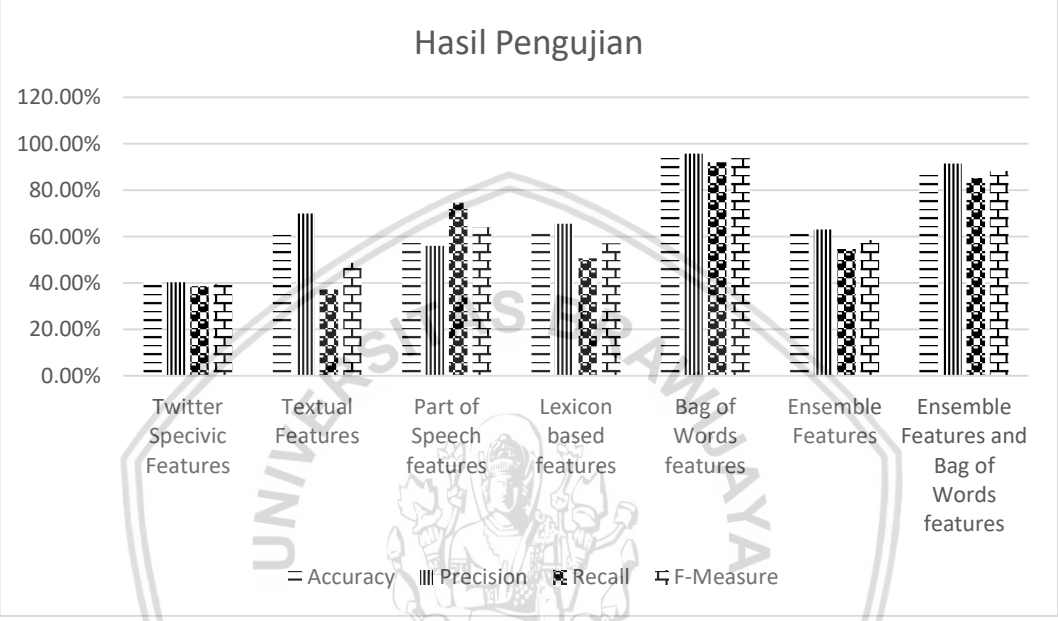
Twitter spesific features yang memiliki kekurangan karena fitur-fiturnya kurang cukup untuk mewakili suatu kelas seperti fitur tanda *username* atau *hashtag*, *Textual features* yang memiliki kelebihan dengan pengecekan emoticon positif dan negatifnya yang dapat mewakili sebuah kelas tertentu namun memiliki kelemahan karena terkadang terdapat tanda di data dengan kelas negatif yang terdeteksi sebagai emoticon positif dan sebaliknya, part of speech features yang menghitung jumlah kata benda atau lainnya berrdasarkan kelas, serta *lexicon based features* yang dapat menentukan kata positif dan negatif berdasarkan daftar kata bersentimen sehingga dapat mewakili kelas tertentu dengan benar namun masih memiliki kekurangan yaitu kata negatif dapat muncul di data dengan kelas positif dan sebaliknya. Kemudian *bag of words features* pun digabung dengan *ensemble features* pada tahap pengujian ini hingga menghasilkan tingkat akurasi yang lebih tinggi daripada sistem yang hanya menggunakan *ensemble features* saja. Hal ini disebabkan karena *bag of words features* terbentuk oleh seluruh kata hasil dari *preprocessing* yang bahkan tidak terdeteksi oleh *ensemble features*, sehingga dapat memperbanyak fitur yang ada dan meningkatkan akurasi sistem. Sebagai contoh ada sebagian kata yang tidak terdeteksi sebagai bagian ensemble features seperti kata 'tidak' sehingga menampilkan akurasi 61,33%. Namun bag of words mampu menutupi kekurangan tersebut karena *bag of words features* juga mendeteksi seluruh kata dari hasil processing seperti kata 'tidak' menjadi bagian dari fiturnya. Sehingga penggabungan *bag of words features* dengan ensemble lebih besar daripada menggunakan *ensemble features* saja yaitu 88,67%.

Namun akurasi dari gabungan *ensemble features* dan *bag of words* ini lebih rendah dari sistem yang hanya memakai *bag of words features* saja yang mendapat akurasi 94,0%. Hal ini disebabkan karena beberapa kesalahan dari *ensemble features* tersebut belum bisa ditangani oleh bag of words features dengan baik. Misalnya seperti tanda ':/' yang merupakan bagian dari link yang tertaut pada dokumen teks, seperti "https://path.com/p/1IKms9". Namun pada *ensemble features* bagian *Textual features* akan mendeteksi tanda tersebut sebagai *emoticon* negatif, padahal tanda tersebut bukan termasuk *emoticon*. Tidak ada proses *data cleaning* untuk menghapus tautan pada *textual features*. Hal ini menyebabkan kesalahan hasil klasifikasi. Sementara *bag of words features* hanya meliputi fitur-fitur kata saja sehingga *bag of words features* kurang bisa memperbaiki hasil klasifikasi data yang disebabkan oleh *Textual features*. Selain itu, ketika semua fitur digabungkan maka nilai akurasi dari menjadi rata-rata semua fitur yang mana batas antar satu fitur dengan fitur yang lain tidak terlalu jauh. Nilai akurasi *Textual features* dan *Twitter specific features* lebih rendah daripada fitur-fitur yang lain ini dikarenakan tergantung datasetnya. *Twitter specific features* jarang muncul di semua data jadi belum bisa mewakili data-data tersebut sehingga fitur tersebut kurang berguna akibatnya fiturnya kurang baik.

Hal ini juga yang menyebabkan nilai *accuracy*, *precision*, *recall*, dan *f-measure* dari sistem yang menggunakan *ensemble features* dan *bag of words* lebih rendah daripada sistem yang menggunakan *bag of words features* saja. Sementara itu, nilai *precision*-nya sebesar 0,9143 dan nilai *recall*-nya sebesar

0,8533 yang berarti tingkat ketepatan analisis sentimen ini tinggi dan memiliki banyak kesesuaian informasi dan didapat dari percobaan serta dari data yang sebenarnya. Sehingga nilai *f-measure* yang didapatkan sebesar 0,8828.

Berikut ini adalah perbandingan hasil pengujian dari seluruh fitur analisis sentimen film pada Twitter berbahasa Indonesia menggunakan *Ensemble Features* dan Naïve Bayes pada Diagram 5.1.



Gambar 5.1 Hasil Pengujian Seluruh Fitur

BAB 6 PENUTUP

Pada bab penutup ini akan menjelaskan tentang kesimpulan dari penelitian yang dilakukan dan saran untuk penelitian selanjutnya.

6.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan sebelumnya, maka dapat diambil kesimpulan mengenal analisis sentimen film pada Twitter berbahasa Indonesia menggunakan *Ensemble Features* dan Naïve Bayes yaitu sebagai berikut:

1. Penggunaan *ensemble features* terhadap hasil analisis sentimen opini film pada Twitter menghasilkan akurasi sebesar 61,33 %, *precision* sebesar 0,6308, *recall* sebesar 0,5467, dan *f-measure* sebesar 0,5857. Hal ini dikarenakan sebagian fitur-fitur tersebut dapat mewakili kelas data dengan baik sehingga sistem dapat menghasilkan keluaran kelas dengan benar. Namun adanya kalimat sindiran dan fitur-fitur yang kurang mewakili kelas suatu data membuat sebagian proses klasifikasi kurang tepat. Kemudian dilakukan penambahan *bag of words features* pada sistem tersebut, maka akurasinya meningkat menjadi 88,67 %, *precision* sebesar 0,9143, *recall* sebesar 0,8533, dan *f-measure* sebesar 0,8828. Hal ini disebabkan oleh *bag of words features* memiliki banyak fitur kata yang dapat mewakili kelas data dan melengkapi sebagian kekuarangan dari *ensemble features* sehingga dapat menghasilkan banyak hasil klasifikasi yang benar.
2. Pada penelitian ini juga dilakukan pengujian terhadap setiap jenis fitur dari *ensemble features* seperti *Twitter specific features, textual features, part of speech features, lexicon based features, bag of words features, ensemble features,* serta *ensemble features* itu sendiri dan *bag of words features*. Di antara fitur tersebut didapatkan bahwa *Bag of Words features* memiliki hasil pengujian tertinggi yaitu akurasi sebesar 94,0 %, *precision* sebesar 0,9583, *recall* sebesar 0,92, dan *f-measure* sebesar 0,9388. Hal ini dikarenakan menggunakan fitur jumlah kata yang sama dari tiap dokumen lebih efektif dari fitur jumlah kata bersentimen, jumlah kata benda, sifat, dll., jumlah tanda pada data Twitter, atau jumlah fitur tekstual. Namun apabila *bag of words features* digabung dengan *ensemble features* menghasilkan akurasi yang lebih rendah dari akurasi sistem yang menggunakan *bag of words* saja karena sebagian kelebihan *bag of words* tertutupi oleh kekurangan *ensembles features* seperti pada kalimat sindiran. Berdasarkan hasil penelitian ini pula ditemukan bahwa terdapat hubungan positif antara *ensemble features* dengan performa klasifikasi.

6.2 Saran

Berdasarkan penelitian yang telah dilakukan sebelumnya terdapat beberapa kekurangan yang perlu diperbaiki atau dikembangkan untuk penelitian selanjutnya. Beberapa saran untuk penelitian selanjutnya antara lain:

1. Dapat menambahkan jumlah data latih agar hasil akurasi yang didapat bisa lebih baik.
2. Dapat menambahkan macam-macam *emoticon* positif dan negatif yang baru ke dalam daftar *emoticon*.
3. Dapat menambahkan macam-macam kata positif dan negatif yang baru ke dalam daftar *sentiwords* dan penanganan kalimat sindiran.
4. Sistem dapat mendeteksi kalimat negasi.
5. Sistem dapat menangani kesalahan dalam mendeteksi *emoticon* pada sebuah tautan.

DAFTAR PUSTAKA

Aggarwal, S. (2013). Naïve Bayes Classifier with Various Smoothing, *4*(April), 873–876.

Feldman, R., & Sanger, J. (2006). *The Text Mining Handbook*. https://doi.org/10.1017/CBO9780511546914

Haddi, E., Liu, X., & Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, *17*, 26–32. https://doi.org/10.1016/j.procs.2013.05.005

Hossin, M., & Sulaiman, M. N. (2015). a Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, *5*(2), 1–11. https://doi.org/10.5121/ijdkp.2015.5201

Issues, C. S. (2010). International Journal of Computer Science Issues. *Online*, *7*(3). https://doi.org/10.1.1.402.9250

John, G. H., & Langley, P. (2013). Estimating Continuous Distributions in Bayesian Classifiers, 338–345. https://doi.org/10.1.1.8.3257

Kontopoulos, E., Berberidis, C., Dergiades, T., & Bassiliades, N. (2013). Ontology-based sentiment analysis of twitter posts. *Expert Systems with Applications*, *40*(10), 4065–4074. https://doi.org/10.1016/j.eswa.2013.01.001

Kwak, H., Lee, C., Park, H., & Moon, S. (2010). What is Twitter, a Social Network or a News Media? Categories and Subject Descriptors. *Www'10*, 591–600. https://doi.org/10.1145/1772690.1772751

Manning, C. D., Raghavan, P., & Schütze, H. (2008). An introduction to information retrieval.

McCallum, A., & Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. *AAAI/ICML-98 Workshop on Learning for Text Categorization*, 41–48. https://doi.org/10.1.1.46.1529

Mollett, A., Moran, D., & Dunleavy, P. (2011). *Using Twitter in university research , teaching and impact activities A guide for academics and researchers*. *Read*. https://doi.org/10.1016/0003-9861(77)90346-0

Patodkar, V. N., & I.R, S. (2016). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *Ijarcce*, *5*(12), 320–322. https://doi.org/10.17148/IJARCCE.2016.51274

Perdana, R. S., Suprapto, & Regasari, R. (2013). Pengkategorian Pesan Singkat Berbahasa Indonesia pada Jejaring Sosial Twitter dengan Metode Klasifikasi Naïve Bayes. *Jurnal PTIIK UB*, (September), 1–12.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation*, *60*(5), 503–520.

https://doi.org/10.1108/00220410410560582

Rossi, A., Lestari, T., Perdana, R. S., & Fauzi, M. A. (2017). Analisis Sentimen Tentang Opini Pilkada Dki 2017 Pada Dokumen Twitter Berbahasa Indonesia Menggunakan Näive Bayes dan Pembobotan Emoji. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(12), 1718–1724.

Serrano-Guerrero, J., Olivas, J. A., Romero, F. P., & Herrera-Viedma, E. (2015). Sentiment analysis: A review and comparative analysis of web services. *Information Sciences*, *311*, 18–38. https://doi.org/10.1016/j.ins.2015.03.040

Siddiqua, U. A., Ahsan, T., & Chy, A. N. (2016). Combining a Rule-based Classifier with Ensemble of Feature Sets and Machine Learning Techniques for Sentiment Analysis on Microblog, 16–21.

Tsai, F. S. (2011). Text mining and visualisation of Protein-Protein Interactions. *International Journal of Computational Biology and Drug Design*, *4*(3), 239. https://doi.org/10.1504/IJCBDD.2011.041412

Vijayarani, S., Ilamathi, J., & Nithya, M. (2015). Preprocessing Techniques for Text Mining - An Overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16. Retrieved from http://www.ijcscn.com/Documents/Volumes/vol5issue1/ijcscn2015050102.pdf